**Задача 1, forks & pipes, type 1­**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 12 либо 132 | Базовый |
| 2 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int fd[2];  pipe(fd);  char x[] = "ab\n";  if(fork()) {  puts(x + 1);  write(fd[1], x, 1);  wait(0);  } else {  wait(0);  write(fd[1], &x[1], 1);  read(fd[0], x, 1);  read(fd[0], x+1, 1);  }  puts(x);  return 0;  } | **b**  **ab**  **ab**  либо  **b**  **ba**  **ab** | Казачук М.А. |
| 3 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int fd[2];  pipe(fd);  char x[] = "01\n";  if(fork()) {  puts(x);  write(fd[1], x, 1);  wait(NULL);  } else {  write(fd[1], &x[1], 1);  read(fd[0], x, 1);  read(fd[0], x+1, 1);  puts(x);  }  return 0;  } | **01**  **01**    либо  **01**  **10** | Казачук М.А. |
| 4 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int fd[2];  pipe(fd);  char x[] = "01\n";  if(fork()) {  write(fd[1], x, 1);  wait(NULL);  } else {  write(fd[1], &x[1], 1);  read(fd[0], x, 1);  read(fd[0], x+1, 1);  }  puts(x);  return 0;  } | **01**  **01**    либо  **10**  **01** | Казачук М.А. |
| 5 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {   pid\_t pid;  int fd[2];  int x = 1;   pipe(fd);   if( (pid = fork()) > 0 ) {   read(fd[0], &x, sizeof(int));   kill(pid, SIGKILL);   wait(NULL);   }   else {   printf("%d", x);   x = 2;   write(fd[1], &x, sizeof(int));   x = 3;   }  printf("%d", x);  return 0;  } | 12 либо 132  форки+пайпы, 1 | Кузина Л.Н. |
| 6 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  printf("%d\n", x);  wait(NULL);  }  else {  printf("%d\n", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  kill(getppid(), SIGKILL);  printf("%d\n", x);  }  return 0;  } | **13**  **123**  **форки+пайпы, 3** | Кузина Л.Н. |
| 7 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  pid\_t pid;  int fd[2];  int x = 0;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  printf("%d", x);  kill(pid, SIGKILL);  wait(NULL);  }  else {  x = 1;  write(fd[1], &x, sizeof(int));  x = 2;  }  printf("%d", x);  return 0;  } | **11**  **121**  **форки+пайпы, 4** | Кузина Л.Н. |
| 8 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  pid\_t pid;  int fd[2];  int x = 3;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x --;  write(fd[1], &x, sizeof(int));  x--;  }  printf("%d", x);  return 0;  } | Ответ: 32 или 312 | Волкова И.А. |
| 9 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  pid\_t pid;  int fd[2];  int x = 5;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  wait(NULL);  }  else {  printf("%d", x);  x +=2;  write(fd[1], &x, sizeof(int));  kill(pid, SIGKILL);  x--;  }  printf("%d", x);  return 0;  } | Ответ: 56 либо 567 | Волкова И.А. |
| 10 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  pid\_t pid;  **int** fd[2];  **int** x = 2;  pipe(fd);  **if** ( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  exit(0);  }  else {  printf("%d", x);  x +=2;  write(fd[1], &x, sizeof(int));  kill(pid, SIGKILL);  x--;  }  printf("%d", x);  return 0;  } | Ответ: 23 | Волкова И.А. |
| 11 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main(void)  {  pid\_t pid;  int fd[2];  int x = 10;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 11;  write(fd[1], &x, sizeof(int));  x = 12;  }  printf("%d", x);  return 0;  } | 1011 либо 101211 | Корныхин Е.В. |
| 12 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main(void)  {  pid\_t pid;  int fd[2];  int x = 3;  pipe(fd);  if( (pid = fork()) == 0 ) {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 1;  }  else {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  printf("%d", x);  return 0;  } | 32 либо 312 | Корныхин Е.В. |
| 13 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main(void)  {  int t = 1;  int fd[2]; pipe(fd);  pid\_t pid = fork();  if (pid) {  read(fd[0], &t, sizeof t);  ++t;  kill(pid, SIGKILL);  wait(NULL);  } else {  printf("%d", t);  t = 2;  write(fd[1], &t, sizeof t);  t = 3;  }  printf("%d", t);  return 0;  } | 13 либо 133 | Корныхин Е.В. |
| 14 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  x=2;  write(fd[1], &x, sizeof(int));  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  read(fd[0], &x, sizeof(int));  x++;  write(fd[1], &x, sizeof(int));  }  printf("%d", x);  return 0;  } | 12 либо 133  форки+пайпы, 1 | Тюляева В.В. |
| 15 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  write(fd[1], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  read(fd[0], &x, sizeof(int));  printf("%d", x);  }  printf("%d", x);  return 0;  } | 1 либо 11 либо 111  форки+пайпы, 1 | Тюляева В.В. |
| 16 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 5;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 1;  write(fd[1], &x, sizeof(int));  x = 2;  }  printf("%d", x);  return 0;  } | 51 либо 521  форки+пайпы, 1 | Гомзин А.Г. |
| 17 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  char x = ‘a’;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(char));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%c", x);  x = ‘c’;  write(fd[1], &x, sizeof(char));  x = ‘f’;  }  printf("%c", x);  return 0;  } | ac либо afc  форки+пайпы, 1 | Гомзин А.Г. |
| 18 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 0;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 1;  }  printf("%d", x);  return 0;  } | 02 либо 012  форки+пайпы, 1 | Гомзин А.Г. |
| 19 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  int fd[2], x;  pid\_t pid;  pipe(fd);  x = 4;    if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 7;  write(fd[1], &x, sizeof(int));  x = 9;  }  printf("%d", x);  return 0;  } | 47 либо 497 | Никольский И.М. |
| 20 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 0;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 7;  write(fd[1], &x, sizeof(int));  x = 9;  }  printf("%d", x);  return 0;  } | 07 либо 097 | Никольский И.М. |
| 21 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 5;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 7;  write(fd[1], &x, sizeof(int));  x = 9;  }  printf("%d", x);  return 0;  } | 57 либо 597 | Никольский И.М. |
| 22 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  pid\_t pid;  **int** pp[2];  **char** c = ‘a’;  pipe(pp);  **if**( (pid = fork()) > 0 ) {  read(pp[0], &c, **sizeof** (c));  kill(pid, SIGKILL);  wait(NULL);  }  **else** {  printf("%c", c);  c = ‘b’;  write(pp[1], &c, **sizeof** (c));  c = ‘c’;  }  printf("%c", c);  **return** 0;  } | ab либо acb  форки+пайпы, 1 | Вылиток А.А. |
| 23 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  pid\_t pid;  **int** pp[2];  **char** c = ‘x’;  pipe(pp);  **if**( (pid = fork()) > 0 ) {  read(pp[0], &c, **sizeof** (c));  kill(pid, SIGKILL);  wait(NULL);  }  **else** {  printf("%c", c);  c = ‘y’;  write(pp[1], &c, **sizeof** (c));  c = ‘c’;  }  printf("%c", c);  **return** 0;  } | xy либо xcy  форки+пайпы, 1 | Вылиток А.А. |
| 24 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  pid\_t pid;  **int** pp[2];  **char** c = ‘5’;  pipe(pp);  **if**( (pid = fork()) > 0 ) {  read(pp[0], &c, **sizeof** (c));  kill(pid, SIGKILL);  wait(NULL);  }  **else** {  printf("%c", c);  c = ‘b’;  write(pp[1], &c, **sizeof** (c));  c = ‘c’;  }  printf("%c", c);  **return** 0;  } | 5b либо 5cb  форки+пайпы, 1 | Вылиток А.А. |
| 25 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 4;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  wait(NULL);  }  else {  printf("%d", x);  x = 5;  write(fd[1], &x, sizeof(int));  x = 6;  }  printf("%d", x);  return 0;  } | 465  форки+пайпы, 1 | Глазкова В.В. |
| 26 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 7;  pipe(fd);  if( (pid = fork()) > 0 ) {  printf("%d", x);  x = 8;  write(fd[1], &x, sizeof(int));  x = 9;  kill(pid, SIGKILL);  wait(NULL);  }  else {  read(fd[0], &x, sizeof(int));  }  printf("%d", x);  return 0;  } | 79 либо 789  форки+пайпы, 1 | Глазкова В.В |
| 27 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 6;  pipe(fd);  if( (pid = fork()) == 0 ) {  read(fd[0], &x, sizeof(int));  }  else {  printf("%d", x);  x = 7;  write(fd[1], &x, sizeof(int));  x = 8;  kill(pid, SIGKILL);  wait(NULL);  }  printf("%d", x);  return 0;  } | 68 либо 678  форки+пайпы, 1 | Глазкова В.В. |
| 28 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int x = 5;  pid\_t pid;  int fd[2];  pipe(fd);  if( pid = fork()) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  } else {  printf("%d", x);  x = 8;  write(fd[1], &x, sizeof(int));  x = 4;  }  printf("%d", x);  return 0;  } | 58 либо 548 | Дряженов А.А. |
| 29 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int x = 5;  pid\_t pid;  int fd[2];  pipe(fd);  if( (pid = fork()) == 0) {  printf("%d", x);  x = 8;  write(fd[1], &x, sizeof(int));  x = 4;  } else {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  printf("%d", x);  return 0;  } | 9. 58 либо 548 | Дряженков А.А. |
| 30 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char x = ‘a’;  pid\_t pid;  int fd[2];  pipe(fd);  if( (pid = fork()) == 0) {  printf("%c", x);  c = ‘b’;  write(fd[1], &x, 1);  c = ‘c’;  } else {  read(fd[0], &x, 1);  kill(pid, SIGKILL);  wait(NULL);  }  printf("%c", x);  return 0;  } | 17. ab либо acb | Дряженков А.А. |
| 31 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int y = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &y, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d,%d \n", y,y);  x = 2;  write(fd[1], &y, sizeof(int));  x = 3;  }  printf("%d", y);  return 0;  } | 1,1  2  либо  1,1  32  форки+пайпы, 1 | Полякова И.Н. |
| 32 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int g[2];  int x = 2;  pipe(g);  if( (pid = fork()) > 0 ) {  read(g[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 1;  write(g[1], &x, sizeof(int));  x = 2;  }  printf("%d", x);  return 0;  } | 21 либо 221  форки+пайпы, 1 | Полякова И.Н. |
| 33 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int h[2];  int x = 1;  pipe(fd);  pipe(h);  if( (pid = fork()) > 0 ) {  write(fd[1], &x, sizeof(int));  read(h[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d\n", x);  x = 2;  write(h[1], &x, sizeof(int));  x = 3;  }  printf("%d\n", x);  return 0;  } | 1  2  либо  1  3  2  форки+пайпы, 1 | Полякова И.Н. |
| 34 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int s = 9;  pipe(fd);  if( (pid = fork()) > 0 ) {  s = 8  read(fd[0], &s, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", s);  s = 7;  write(fd[1], &s, sizeof(int));  s = 6;  }  printf("%d", s);  return 0;  } | 97 либо 967 | Жуков К.А. |
| 35 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  short b = 3;  pipe(fd);  if ((pid = fork()) == 0) {  printf("%d", b);  b = 4;  write(fd[1], &b, sizeof(short));  b = 5;  }  else {  b = 6  read(fd[0], &b, sizeof(short));  kill(pid, SIGKILL);  wait(NULL);  }  printf("%d", b);  return 0;  } | 34 либо 354 | Жуков К.А. |
| 36 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  char c = ‘a’;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &c, sizeof(char));  kill(pid, SIGKILL);  wait(NULL);  }  else {  putchar(c);  c = ‘b’;  write(fd[1], &c, sizeof(char));  c = ‘d’;  }  putchar(c);  return 0;  } | ab либо adb | Жуков К.А. |
| 37 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  close(fd[1]);  kill(pid, SIGKILL);  read(fd[0], &x, sizeof(int));  wait(NULL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 1 либо 11 либо 12 либо 132  форки+пайпы, 1 | Шестимеров А.А. |
| 38 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  close(fd[1]);  kill(pid, SIGKILL);  read(fd[0], &x, sizeof(int));  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  wait(NULL);  return 0;  } | 1 или 11 или 12 или 132 или 123  форки+пайпы, 1 | Шестимеров А.А. |
| 39 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  close(fd[1]);  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  wait(NULL);  return 0;  } | 12 либо 132 либо 123  форки+пайпы, 1 | Шестимеров А.А. |
| 40 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork())== 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 13 либо 1  форки+пайпы, 1 | Сучков Е.П. |
| 41 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  ++x;  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 22 или 232 | Сучков Е.П. |
| 42 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) == 0 ) {  read(fd[0], &x, sizeof(int));  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  x = 3;  }  printf("%d", x);  return 0;  } | 13 или 123 | Сучков Е.П. |
| 43 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  kill(pid, SIGKILL);  read(fd[0], &x, sizeof(int));  wait(NULL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 12 либо 132 либо 1 либо 11  форки+пайпы, 1 | Головин И.Г. |
| 44 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  exit(0);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 1 либо 13  форки+пайпы, 1 | Головин И.Г. |
| 45 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  exit(0);  }  printf("%d", x);  return 0;  } | 1 либо 12 - сложность - max  форки+пайпы, 1 | Головин И.Г. |
| 46 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%d", x);  x--;  write(fd[1], &x, sizeof(int));  x++;  }  printf("%d", x);  return 0;  } | 10 либо 110  форки+пайпы, 1 | Семенов А.Н. |
| 47 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  int x = 1;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(int));  printf("%d", x);  wait(NULL);  x = 4;  }  else {  printf("%d", x);  x = 2;  write(fd[1], &x, sizeof(int));  x = 3;  }  printf("%d", x);  return 0;  } | 1234 либо 1324  форки+пайпы, 1 | Семенов А.Н. |
| 48 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  pid\_t pid;  int fd[2];  char p = ‘q’;  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &p, 1));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%c", p);  p = ‘w’;  write(fd[1], &x,1);  p = ‘e’;  }  printf("%c", p);  return 0;  } | qw либо qew  форки+пайпы, 1 | Семенов А.Н. |
| 49 | **#**include <stdio.h>  #include <unistd.h>  #include <signal.h>  #include <sys/types.h>  #include <sys/wait.h>  int main()  {  pid\_t pid;  int fd[2];  char x = 'x';  pipe(fd);  if( (pid = fork()) > 0 ) {  read(fd[0], &x, sizeof(x));  kill(pid, SIGCHLD);  wait(NULL);  }  else {  printf("%c", x);  x = 'y';  write(fd[1], &x, sizeof(x));  x = 'z';  }  printf("%c", x);  return 0;  } | /\*  \* answer: 'xzy'  \*/ | Чернов А.В. |
| 50 | #include <stdio.h>  #include <unistd.h>  #include <signal.h>  #include <sys/types.h>  #include <sys/wait.h>  int main()  {  pid\_t pid;  int fd[2];  char x = 'x';  pipe(fd);  if( (pid = fork()) > 0 ) {  printf("%c", ++x);  read(fd[0], &x, sizeof(x));  kill(pid, SIGKILL);  wait(NULL);  }  else {  printf("%c", x);  ++x;  write(fd[1], &x, sizeof(x));  ++x;  }  printf("%c", x);  return 0;  } | /\*  \* answer: 'yxy', 'yxy', 'xyy'  \*/ | Чернов А.В. |
| 51 | #include <stdio.h>  #include <unistd.h>  #include <signal.h>  #include <sys/types.h>  #include <sys/wait.h>  int main()  {  pid\_t pid;  int fd[2];  char x = 'x';  pipe(fd);  if( (pid = fork()) > 0 ) {  printf("%c", ++x);  read(fd[0], &x, sizeof(x));  kill(pid, SIGKILL);  wait(NULL);  }  else {  ++x;  write(fd[1], &x, sizeof(x));  usleep(1000);  ++x;  }  printf("%c", x);  return 0;  } | /\*  \* answer: 'yy', 'zyy'  \*/ | Чернов А.В. |

**Задача 2, forks & pipes, type 2**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);­  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | acbdf либ­­­о abcdf либо bacdf | Базовый |
| 2 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int fd[2];  pipe(fd);  char x[] = "123\n";  if(fork()) {  puts(x + 1);  write(fd[1], x+2, 1);  wait(NULL);  }  else {  write(fd[1], &x[0], 1);  read(fd[0], x, 1);  read(fd[0], x+1, 1);  }  puts(x);  return 0;  } | **23**  **313**  **123**  либо  **23**  **133**  **123** | Казачук М.А. |
| 3 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int fd[2];  pipe(fd);  char x[] = "qw\n";  if(fork()) {// pid=A  write(fd[1], x, 1);  wait(NULL);  }  else { //pid=B  write(fd[1], &x[1], 1);  read(fd[0], x, 1);  read(fd[0], x+1, 1);  }  printf("%d " ,getpid());  puts(x);  return 0;  } | **B qw**  **A qw**  либо  **B wq**  **A qw** | Казачук М.А. |
| 4 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main() {  int fd[2];  pipe(fd);  char x[] = "12\n";  if(fork()) {  puts(x + 1);  write(fd[1], x, 1);  wait(NULL);  }  else {  write(fd[1], &x[1], 1);  read(fd[0], x, 1);  read(fd[0], x+1, 1);  }  puts(x);  return 0;  } | **2**  **12**  **12**  либо  **2**  **21**  **12** | Казачук М.А. |
| 5 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbdf либо abcdf либо bacdf**  форки+пайпы, 2 | Кузина Л.Н. |
| 6 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd1[2], fd2[2];  pipe(fd1);  pipe(fd2);  if(fork() == 0) {  puts("1");  write(fd1[1], &c, 1);  read(fd2[0], &c, 1);  puts("2");  exit(0);  }  else {  puts("3");  read(fd1[0], &c, 1);  puts("4");  write(fd2[1], &c, 1);  wait(NULL);  }  puts("5");  return 0;  } | **форки+пайпы, 5**  3 1 4 2 5  1 3 4 2 5 | Кузина Л.Н. |
| 7 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd1[2], fd2[2];  pipe(fd1);  pipe(fd2);  if(fork() == 0) {  puts("1");  write(fd1[1], &c, 1);  read(fd2[0], &c, 1);  puts("2");  exit(0);  }  else {  puts("3");  read(fd1[0], &c, 1);  write(fd2[1], &c, 1);  puts("4");;  wait(NULL);  }  puts("5\n");  return 0;  } | **форки+пайпы, 6**  3 1 4 2 5  3 1 2 4 5  1 3 4 2 5  1 3 2 4 5 | Кузина Л.Н. |
| 8 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Int main(){  **char** c;  **int** ff[2], pp[2];  pipe (ff);  pipe (pp);  **if** (fork() == 0) {  write (ff [1], &c, 1);  putchar ('1');  read (pp[0], &c, 1);  putchar ('7');  exit (0);  }  putchar ('3');  read (ff [0], &c, 1);  putchar ('5');  write (pp[1], &c, 1);  wait (NULL);  putchar ('9');  **return** 0;  } | **35179 либо 31579 либо 13579** | Волкова И.А. |
| 9 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Int main(){  **char** c;  **int** ff[2], pp[2];  pipe (ff);  pipe (pp);  **if** (fork() == 0) {  write (ff [1], &c, 1);  putchar ('1');  read (pp[0], &c, 1);  read (ff [0], &c, 1);  putchar ('7');  }  write (pp[1], &c, 1);  putchar ('9');  **return** 0;  } | **9179 либо 1799 либо 1979** | Волкова И.А. |
| 10 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Int main(){  **char** c;  **int** ff[2], pp[2];  pipe (ff);  pipe (pp);  **if** (fork() == 0) {  write (ff [1], &c, 1);  close (pp[1]);  close (ff[1]);  read (pp[0], &c, 1);  putchar ('1');  read (ff [0], &c, 1);  putchar ('7');  }  read (ff [0], &c, 1);  putchar ('2');  close (pp[1]);  **return** 0;  } | Ответ: **2172** | Волкова И.А. |
| 11 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main(void)  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('p');  read(fd2[0], &c, 1);  putchar('q');  exit(0);  }  putchar('r');  read(fd[0], &c, 1);  putchar('s');  write(fd2[1], &c, 1);  wait(NULL);  putchar('t');  return 0;  } | **rspqt либо rpsqt либо prsqt** | Корныхин Е.В. |
| 12 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main(void)  {  char c[1];  int fd1[2]; pipe(fd1);  int fd2[2]; pipe(fd2);  if(fork() == 0) {  write(fd1[1], c, 1);  putchar('c');  read(fd2[0], c, 1);  putchar('a');  } else {  putchar('a');  read(fd1[0], c, 1);  putchar('b');  write(fd2[1], c, 1);  wait(NULL);  putchar('b');  }  return 0;  } | **acbab либо abcab либо cabab** | Корныхин Е.В. |
| 13 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main(void)  {  int t = 0, f[2], d[2];  pipe(f);  pipe(d);  if(fork()) {  printf("a");  read(f[0], &t, sizeof t);  write(d[1], &t, sizeof t);  printf("c");  wait(NULL);  printf("f");  exit(0);  }  write(f[1], &t, sizeof t);  printf("b");  read(d[0], &t, sizeof t);  printf("d");  return 0;  } | **acbdf либо abcdf либо abdcf или bacdf или badcf** | Корныхин Е.В. |
| 14 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  putchar('a');  wait(NULL);  return 0;  } | **bcda** или **bcad** или **cbda** или **cbad** или **cabd**  форки+пайпы, 2 | Тюляева В.В. |
| 15 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  return 0;  } | **bcd** или **cbd**  форки+пайпы, 2 | Тюляева В.В. |
| 16 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('x');  read(fd2[0], &c, 1);  putchar('y');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('z');  return 0;  } | **acxyz либо axcyz либо xacyz**  форки+пайпы, 2 | Гомзин А.Г. |
| 17 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('2');  read(fd2[0], &c, 1);  putchar('5');  exit(0);  }  putchar('1');  read(fd[0], &c, 1);  putchar('3');  write(fd2[1], &c, 1);  wait(NULL);  putchar('4');  return 0;  } | **13254 либо 12354 либо 21354**  форки+пайпы, 2 | Гомзин А.Г. |
| 18 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('z');  read(fd2[0], &c, 1);  putchar('4');  exit(0);  }  putchar('x');  read(fd[0], &c, 1);  putchar('a');  write(fd2[1], &c, 1);  wait(NULL);  putchar('9');  return 0;  } | **xaz49 либо xza49 либо zxa49**  форки+пайпы, 2 | Гомзин А.Г. |
| 19 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];    pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('y');  read(fd2[0], &c, 1);  putchar('p');  exit(0);  }  putchar('t');  read(fd[0], &c, 1);  putchar('x');  write(fd2[1], &c, 1);  wait(NULL);  putchar('h');  return 0;  } | **txyph либо tyxph либо ytxph** | Никольский И.М. |
| 20 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('p');  read(fd2[0], &c, 1);  putchar('g');  exit(0);  }  putchar('y');  read(fd[0], &c, 1);  putchar('f');  write(fd2[1], &c, 1);  wait(NULL);  putchar('j');  return 0;  } | **yfpgj либо ypfgj либо pyfgj** | Никольский И.М. |
| 21 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('k');  read(fd2[0], &c, 1);  putchar('g');  exit(0);  }  putchar('p');  read(fd[0], &c, 1);  putchar('v');  write(fd2[1], &c, 1);  wait(NULL);  putchar('t');  return 0;  } | **pvkgf либо pkvgf либо kpvgf** | Никольский И.М. |
| 22 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  **int** x=1;  **int** pp1[2], pp2[2];  pipe(pp1);  pipe(pp2);  **if**(fork() == 0) {  write(pp1[1], &x, **sizeof** (x));  putchar('a');  read(pp2[0], &x, **sizeof** (x));  putchar('b');  exit(0);  }  putchar('1');  read(pp1[0], &x, **sizeof** (x));  putchar('2');  write(pp2[1], &x, **sizeof** (x));  wait(NULL);  putchar('3');  **return** 0;  } | 12ab3 либо 1a2b3 либо a12b3  форки+пайпы, 2 | Вылиток А.А. |
| 23 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  **int** x=1;  **int** pp1[2], pp2[2];  pipe(pp1);  pipe(pp2);  **if**(fork() == 0) {  write(pp1[1], &x, **sizeof** (x));  putchar('a');  read(pp2[0], &x, **sizeof** (x));  putchar('b');  exit(0);  }  putchar('3');  read(pp1[0], &x, **sizeof** (x));  putchar('2');  write(pp2[1], &x, **sizeof** (x));  wait(NULL);  putchar('1');  **return** 0;  } | 32ab1 либо 3a2b1 либо a32b1  форки+пайпы, 2 | Вылиток А.А. |
| 24 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  **int** x=1;  **int** pp1[2], pp2[2];  pipe(pp1);  pipe(pp2);  **if**(fork() == 0) {  write(pp1[1], &x, **sizeof** (x));  putchar('b');  read(pp2[0], &x, **sizeof** (x));  putchar('a');  exit(0);  }  putchar('1');  read(pp1[0], &x, **sizeof** (x));  putchar('2');  write(pp2[1], &x, **sizeof** (x));  wait(NULL);  putchar('3');  **return** 0;  } | 12ba3 либо 1b2a3 либо b12a3  форки+пайпы, 2 | Вылиток А.А. |
| 25 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c=‘e';  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acdf**  форки+пайпы, 2 | Глазкова В.В. |
| 26 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c=‘a’;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  putchar(‘f');  wait(NULL);  return 0;  } | **acbdf либо abcdf либо bacdf либо**  **acbfd либо abcfd либо bacfd**  форки+пайпы, 2 | Глазкова В.В. |
| 27 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c=‘6’;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('2');  read(fd2[0], &c, 1);  putchar('4');  exit(0);  }  putchar('1');  read(fd[0], &c, 1);  putchar('3');  write(fd2[1], &c, 1);  wait(NULL);  return 0;  } | **1324 либо 1234 либо 2134**  форки+пайпы, 2 | Глазкова В.В. |
| 28 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(!fork()) {  write(fd[1], &c, 1);  putchar('5');  read(fd2[0], &c, 1);  putchar('3');  exit(0);  }  putchar('7');  read(fd[0], &c, 1);  putchar('9');  write(fd2[1], &c, 1);  wait(NULL);  putchar('0');  return 0;  } | 3. 79530 либо 75930 либо 57930 | Дряженков А.А. |
| 29 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork()) {  putchar('7');  read(fd[0], &c, 1);  putchar('9');  write(fd2[1], &c, 1);  wait(NULL);  putchar('3');  exit(0);  }  write(fd[1], &c, 1);  putchar('5');  read(fd2[0], &c, 1);  putchar('0');  return 0;  } | 11. 79503 либо 75903 либо 57903 | Дряженков А.А. |
| 30 | Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = ‘2’;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork()) {  putchar('7');  read(fd[0], &c, 1);  putchar('5');  write(fd2[1], &c, 1);  wait(NULL);  putchar('3');  } else {  write(fd[1], &c, 1);  putchar('9');  read(fd2[0], &c, 1);  putchar('0');  }  return 0;  } | 19. 75903 либо 79503 либо 97503 | Дряженков А.А. |
| 31 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  char m= ‘A’;  int fd[2], fd2[2], fd3[2];  pipe(fd);  pipe(fd2);  pipe(fd3);  if(fork() == 0) {  write(fd3[1], &m, 1);  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  write(fd3[1], &m, 1);  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  return 0;  } | **acbd либо abcd либо bacd**  форки+пайпы, 2 | Полякова И.Н. |
| 32 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('d');  read(fd2[0], &c, 1);  putchar('f');  exit(0);  }  putchar('b');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('a');  return 0;  } | **bcdfa либо bdcfa либо dbcfa**  форки+пайпы, 2 | Полякова И.Н. |
| 33 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int g[2], g2[2];  pipe(g);  pipe(g2);  if(fork() == 0) {  write(g[1], &c, 1);  putchar('1');  read(g2[0], &c, 1);  putchar('8');  exit(0);  }  putchar('4');  read(g[0], &c, 1);  putchar('2');  write(g2[1], &c, 1);  wait(NULL);  putchar('6');  return 0;  } | **42186 либо 41286 либо 14286**  форки+пайпы, 2 | Полякова И.Н |
| 34 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char e;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if ( fork() > 0)  {  putchar('z');  read(fd[0], &e, 1);  putchar('y');  write(fd2[1], &e, 1);  wait(NULL);  putchar('x');  exit(0);  }  write(fd[1], &e, 1);  putchar('u');  read(fd2[0], &e, 1);  putchar('v');  return 0;  } | **zyuvx либо zuyvx либо uzyvx** | Жуков К.А. |
| 35 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int x;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &x, sizeof(int));  printf(“2”);  read(fd2[0], &x, sizeof(int));  printf(“4”);  exit(0);  }  printf(“1”);  read(fd[0], &x, sizeof(int));  printf(“3”);  write(fd2[1], &x, sizeof(int));  wait(NULL);  printf(“5”);  return 0;  } | **13245 либо 12345либо 21345** | Жуков К.А. |
| 36 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  int x;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if (fork() > 0) {  printf(“5”);  read(fd[0], &x, sizeof(int));  printf(“3”);  write(fd2[1], &x, sizeof(int));  wait(NULL);  printf(“1”);  exit(0);  }  write(fd[1], &x, sizeof(int));  printf(“4”);  read(fd2[0], &x, sizeof(int));  printf(“2”);    return 0;  } | **53421 либо 54321 либо 45321** | Жуков К.А. |
| 37 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = ‘h’;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  read(fd[0], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  write(fd[1], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbdf либо abcdf**  форки+пайпы, 2 | Шестимеров А.А. |
| 38 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  read(fd[0], &c, 1);  putchar('b');  write(fd2[1], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  write(fd[1], &c, 1);  putchar('c');  read(fd2[0], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbdf либо abcdf либо abdсf**  форки+пайпы, 2 | Шестимеров А.А. |
| 39 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  putchar('h');  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **ahcbdf либо habcdf либо hbacdf либо hacbdа либо ahbcdf**  форки+пайпы, 2 | Шестимеров А.А. |
| 40 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  read(fd[0], &c, 1);  putchar('b');  write(fd2[1], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  write(fd[1], &c, 1);  putchar('c');  read(fd2[0], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbdf либо abcdf либо abdcf**  форки+пайпы, 2 | Сучков Е.П. |
| 41 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() > 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  wait(NULL);  }  else{  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  putchar('f');  exit(0);  }  return 0;  } | **bacfd либо abcfd либо abcdf** | Сучков Е.П. |
| 42 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() > 0) {  read(fd[0], &c, 1);  putchar('b');  write(fd2[1], &c, 1);  putchar('d');  wait(NULL);  }  else{  putchar('a');  write(fd[1], &c, 1);  putchar('c');  read(fd2[0], &c, 1);  putchar('f');  exit(0);  }  return 0;  } | **acbdf либо abdcf либо acbfd** | Сучков Е.П. |
| 43 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = 'x';  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar(c);  read(fd2[0], &c, 1);  putchar(c);  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **xacxf либо axcxf либо acxxf**  форки+пайпы, 2 | Головин И.Г. |
| 44 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c='x';  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() > 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbdf либо abcdf либо bacdf либо bdacf либо badcf** форки+пайпы, 2 | Головин И.Г. |
| 45 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2];  pipe(fd);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd[0], &c, 1);  putchar('d');  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  write(fd[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbdf либо abcdf либо bacdf либо bda либо bad либо abd**  форки+пайпы, 2 | Головин И.Г. |
| 46 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('a');  read(fd2[0], &c, 1);  putchar('b');  exit(0);  }  putchar('c');  read(fd[0], &c, 1);  putchar('d');  write(fd2[1], &c, 1);  wait(NULL);  putchar('e');  return 0;  } | **cdabe либо cadbe либо acdbe**  форки+пайпы, 2 | Семенов А.Н. |
| 47 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar('d');  exit(0);  }  putchar('с');  read(fd[0], &c, 1);  putchar('a');  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **cabdf либо cbadf либо bcadf**  форки+пайпы, 2 | Семенов А.Н. |
| 48 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.** Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = ‘c’;  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('b');  read(fd2[0], &c, 1);  putchar(c);  exit(0);  }  putchar('a');  read(fd[0], &c, 1);  putchar('c');  c = ‘a’;  write(fd2[1], &c, 1);  wait(NULL);  putchar('f');  return 0;  } | **acbaf либо abcaf либо bacaf**  форки+пайпы, 2 | Семенов А.Н. |
| 49 | #include <stdio.h>  #include <unistd.h>  #include <sys/wait.h>  #include <stdlib.h>  int main()  {  char c = '0';  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('1');  read(fd2[0], &c, 1);  putchar('2');  exit(0);  }  putchar('3');  read(fd[0], &c, 1);  putchar('4');  write(fd2[1], &c, 1);  wait(NULL);  putchar('5');  return 0;  } | /\*  \* answer: '34125', '13425', '34125'  \*/ | Чернов А.В. |
| 50 | #include <stdio.h>  #include <unistd.h>  #include <sys/wait.h>  #include <stdlib.h>  int main()  {  char c = '0';  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  putchar('1');  write(fd[1], &c, 1);  putchar('2');  read(fd2[0], &c, 1);  putchar('3');  exit(0);  }  read(fd[0], &c, 1);  putchar('4');  write(fd2[1], &c, 1);  wait(NULL);  putchar('5');  return 0;  } | /\*  \* answer: '14235', '12435'  \*/ | Чернов А.В. |
| 51 | #include <stdio.h>  #include <unistd.h>  #include <sys/wait.h>  #include <stdlib.h>  int main()  {  char c = '0';  int fd[2], fd2[2];  pipe(fd);  pipe(fd2);  if(fork() == 0) {  write(fd[1], &c, 1);  putchar('1');  putchar('2');  read(fd2[0], &c, 1);  putchar('3');  exit(0);  }  read(fd[0], &c, 1);  putchar('4');  write(fd2[1], &c, 1);  wait(NULL);  putchar('5');  return 0;  } | /\*  \* answer: '14235', '12435', '41235'  \*/ | Чернов А.В. |

**Задача 3, semaphores, type 1**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:  1) abbac  2) ababc  3) aabcb  4) aacbb  5) aabbc  6) cabab | Невозможные варианты вывода:  1) abbac  4) aacbb | Базовый |
| 2 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flg = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('2');  putchar('3');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flg = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:  1) 23231  2) 23321  3) 22313  4) 22331  5) 12323  6) 22133  7) 21323 | Невозможные варианты вывода:  2) 23321  6) 22133 | Казачук М.А. |
| 3 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  struct sembuf op; op.sem\_num = 0; op.sem\_flg = 0; op.sem\_op = -1; semop(semId, &op, 1); putchar('2');  putchar('1'); op.sem\_op = 1; semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op; op.sem\_num = 0; op.sem\_flg = 0; op.sem\_op = -1; semop(semId, &op, 1); putchar('3');  op.sem\_op = 1; semop(semId, &op, 1);  Варианты вывода:  1) 21123  2) 21213  3) 22131  4) 22311  5) 22113  6) 32121 | Невозможные варианты вывода:  1) 21123  4) 22311 | Казачук М.А. |
| 4 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flg = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flg = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('3');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:  1) 12213  2) 12123  3) 11232  4) 11322  5) 11223  6) 31212 | Невозможные варианты вывода:  1) 12213  4) 11322 | Казачук М.А. |
| 5 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (С):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:   1. abbac 2. ababc 3. aabcb 4. aacbb 5. aabbc 6. cabab | **Невозможные варианты вывода:**  **1) abbac**  **4) aacbb**  Семафоры 1 | Кузина Л.Н. |
| 6 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено. **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: **semctl (semId, 0, SETVAL, 2);**  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('3');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **С**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  1) 32231  2) 32321  3) 33212  4) 33122  5) 33221  6) 13232 | Ответ: невозможны 1) и 4) | Волкова И.А. |
| 7 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено. **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: **semctl (semId, 0, SETVAL, 3);**  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('3');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **С**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  1) 32231  2) 32321  3) 33212  4) 33122  5) 33221  6) 13232 | Ответ: Невозможен 1 | Волкова И.А. |
| 8 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено. **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: **semctl (semId, 0, SETVAL, 1)**;  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('3');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **С**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  1) 32231  2) 32321  3) 33212  4) 33122  5) 33221  6) 13232 | Ответ: невозможны 1),3), 4) и 5) | Волкова И.А. |
| 9 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 4);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. aabcb 2. aacbb 3. aabac 4. abbac 5. abacb 6. cabab | **Невозможные варианты вывода:**  **2) aacbb**  **3) aabac 4) abbac** | Корныхин Е.В. |
| 10 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (X,Y,Z):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 3);  Данный фрагмент выполняется двумя параллельными процессами (**X и Y**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('b');  putchar('a');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**Z**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 2;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. babac 3. cbaab 4. cbaba 5. bbaca 6. bbcaa 7. bbaac | **Невозможные варианты вывода:**  **1) abbac**  **3) cbaab**  **6) bbcaa** | Корныхин Е.В. |
| 11 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. ababc 3. aabcb 4. aacbb 5. aabbc 6. caabb 7. cabab | **Невозможные варианты вывода:**  **1) abbac 3) aabcb**  **4) aacbb 6) caabb** | Корныхин Е.В. |
| 12 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 3);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. abcab 2. acabb 3. acbab 4. cabab 5. caabb 6. ababc | **Невозможные варианты вывода:**  **2) acabb**  **5) caabb**  Семафоры 1 | Тюляева В.В. |
| 13 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**С**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. ababc 2. abbac 3. aabbc 4. aabcb 5. cabab 6. aacbb | **Невозможные варианты вывода:**  **2) abbac**  **6) aacbb**  Семафоры 1 | Гомзин А.Г. |
| 14 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('x');  putchar('z');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('y');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. xxzzy 2. yxzxz 3. xxzyz 4. xxyzz 5. xzzxy 6. xzxzy | **Невозможные варианты вывода:**  **4) xxyzz**  **5) xzzxy**  Семафоры 1 | Гомзин А.Г. |
| 15 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('x');  putchar('z');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('y');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. xxzzy 2. xxzyz 3. xxyzz 4. xzxzy 5. xzzxy 6. yxzxz | **Невозможные варианты вывода:**  **3) xxyzz**  **5) xzzxy**  Семафоры 1 | Гомзин А.Г. |
| 16 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('x');  putchar('y');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('z');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. xyyxz 2. xyxyz 3. xxyzy 4. xxzyy 5. xxyyz 6. zxyxy | **Невозможные варианты вывода:**  **1) xyyxz**  **4) xxzyy** | Никольский И.М. |
| 17 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('p');  putchar('q');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('r');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. pqqpr 2. pqpqr 3. ppqrq 4. pprqq 5. ppqqr 6. rpqpq | **Невозможные варианты вывода:**  **1) pqqpr**  **4) pprqq** | Никольский И.М. |
| 18 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('3');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. 12213 2. 12123 3. 11232 4. 11322 5. 11223 6. 31212 | **Невозможные варианты вывода:**  **1) 12213**  **4) 11322** | Никольский И.М. |
| 19 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A, B, C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  **struct** sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом C:  **struct** sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:  1221c  1212c  112c2  11c22  1122c  c1212 | Невозможные варианты вывода:  1) 1221c  4) 11c22  Семафоры, 1 | Вылиток А.А. |
| 20 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A, B, C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  **struct** sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом C:  **struct** sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('d');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:  1212d  112d2  11d22  1122d  d1212  1221d | Невозможные варианты вывода:  1) 1221d  6) 11d22  Семафоры, 1 | Вылиток А.А. |
| 21 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A, B, C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (A и B):  **struct** sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('2');  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом C:  **struct** sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  Варианты вывода:  2112c  22c11  2121c  221c1  2211c  c2121 | Невозможные варианты вывода:  1) 2112c  2) 22c11  Семафоры, 1 | Вылиток А.А. |
| 22 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Перечислите варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 4);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('1');  putchar(‘2');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **С:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('3');  op.sem\_op = 2;  semop(semId, &op, 1);  **Варианты вывода:**   1. 12213 2. 12123 3. 11232 4. 11322 5. 11223 6. 31212 | **Невозможные варианты вывода:**  **1) 12213**  **4) 11322**  Семафоры 1 | Глазкова В.В. |
| 23 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 4);  Данный фрагмент выполняется двумя параллельными процессами (**B и C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом A:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 2;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. ababc 3. aabcb 4. aacbb 5. aabbc 6. cabab | **Невозможные варианты вывода:**  **1) abbac**  **4) aacbb**  Семафоры 1 | Глазкова В.В. |
| 24 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 3;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом C:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 3;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. ababc 3. aacbb 4. aabcb 5. aabbc 6. cabab | **Невозможные варианты вывода:**  **1) abbac**  **3) aacbb**  Семафоры 1 | Глазкова В.В. |
| 25 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  putchar('2');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('3');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  12213  12123  11232  11322  11223  31212 | 2. Невозможные варианты вывода:  1) 12213  4) 11322 | Дряженков А.А. |
| 26 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  putchar('x');  putchar('y');  op.sem\_op = 4;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  putchar('z');  op.sem\_op = 4;  semop(semId, &op, 1);  **Варианты вывода:**  xyyxz  xyxyz  xxyzy  zyxxy  xxzyy  xxyyz  zxyxy | 10. Невозможные варианты вывода:  1) xyyxz  4) zyxxy  5) xxzyy | Дряженков А.А. |
| 27 | Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  putchar('0');  putchar('1');  op.sem\_op = 4;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом (**C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  putchar('2');  op.sem\_op = 4;  semop(semId, &op, 1);  **Варианты вывода:**  01102  01012  00121  21001  00211  00112  20101 | 18. Невозможные варианты вывода:  1) 01102  4) 21001  5) 00211 | Дряженков А.А. |
| 28 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('b');  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  bccba  bcbca  bbcac  bbacc  bbcca  abcbc | **Невозможные варианты вывода:**  **1) bccba**  **4) bbacc**  Семафоры 1 | Полякова И.Н. |
| 29 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('6');  putchar('4');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('5');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  1) 64465  2) 64645  3) 66454  4) 66544  5) 66445  6) 56464 | **Невозможные варианты вывода:**  **1) 64465**  **4) 66544**  Семафоры 1 | Полякова И.Н. |
| 30 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('x');  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('y');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  1) x11xy  2) x1x1y  3) xx1y1  4) xxy11  5) xx11y  6) yx1x1 | **Невозможные варианты вывода:**  **1) x11xy**  **4) xxy11**  Семафоры 1 | Полякова И.Н. |
| 31 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  printf(“1”);  printf(“2”);  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **C**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  printf(“3”);  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. 12213 2. 12123 3. 11232 4. 11322 5. 11223 6. 31212 | **Невозможные варианты вывода:**  **1) 12213**  **4) 11322** | Жуков К.А. |
| 32 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено. Напишите варианты вывода (написаны ниже), возможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A** **и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0; op.sem\_op = -1; semop(semId, &op, 1); putchar('t'); putchar('u'); op.sem\_op = 1; semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **С**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0; op.sem\_op = -1; semop(semId, &op, 1); putchar('w'); op.sem\_op = 1; semop(semId, &op, 1);  1) ttuuw  2) wtutu  3) ttuwu  4) tuutw  5) tutuw  6) ttwuu | **tutuw либо ttuwu либо ttuuw либо wtutu** | Жуков К.А. |
| 33 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 4);  Данный фрагмент выполняется одним параллельным процессом **A**:  struct sembuf op;  op.sem\_flag = 0;  op.sem\_num = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('x');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется двумя параллельными процессами (**B и C**):  struct sembuf op;  op.sem\_flag = 0;  op.sem\_num = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('y');  putchar('z');  op.sem\_op = 2;  semop(semId, &op, 1);  **Варианты вывода:**   1. yzzyx 2. yzyzx 3. yyzxz 4. yyxzz 5. yyzzx 6. xyzyz | **Невозможные варианты вывода:**  **1) yzzyx**  **4) yyxzz** | Жуков К.А. |
| 34 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0,  SETVAL, 3);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. ababc 3. aabcb 4. aacbb 5. aabbc 6. cabab | **Невозможные варианты вывода:**  abbac  aabcb  aacbb  aabbc  Семафоры 1 | Шестимеров А.А. |
| 35 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0,  SETVAL, 3);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом C:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 2;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. ababc 3. aabcb 4. aacbb 5. aabbc 6. cabab | **Невозможные варианты вывода:**  **1) abbac**  Семафоры 1 | Шестимеров А.А. |
| 36 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0,  SETVAL, 3);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**   1. abbac 2. ababc 3. aabcb 4. aacbb 5. aabbc 6. cabab | **Невозможные варианты вывода:**  abbac  ababc  aabcb  aacbb  aabbc  cabab  Семафоры 1 | Шестимеров А.А. |
| 37 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 2;  semop(semId, &op, 1);  **Варианты вывода:**  abbac  ababc  acbab  aabcb  aacbb  aabbc  cabab | **Невозможные варианты вывода:**  **1) abbac**  **3) acbab**  **4) aabcb**  **5) aacbb**  Семафоры 1 | Сучков Е.П. |
| 38 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова:  semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  abcab  abbac  ababc  aabcb  aacbb  aabbc  cabab | **abbac**  **4) aabcb**  **aacbb**  **aabbc** | Сучков Е.П. |
| 39 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова:  semctl(semId, 0, SETVAL, 3);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 2;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  abbac  ababc  abcab  aabcb  aacbb  aabbc  cabab  acbab | **abbac**  **4) aabcb**  **5) aacbb**  **6) aabbc** | Сучков Е.П. |
| 40 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении заданного фрагмента программы тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется тремя параллельными процессами (**A, B, С**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  abbaab  aaabbb  ababab  aabbab  aabbba | Невозможные варианты вывода:  1) abbaab  2) aaabbb  5) aabbba  Семафоры 1 | Головин И.Г. |
| 41 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом **C**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  abbac  ababc  aabcb  aacbb  aabbc  cabab | **Невозможные варианты вывода:**  **1) abbac**  **4) aacbb**  Семафоры 1 | Головин И.Г. |
| 42 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется одним параллельным процессом **A**:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется двумя параллельными процессами (**B и C**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  cabc  abcc  ccab  cacb  cbac  acbc | **Невозможные варианты вывода:**  **5) cbac**  Семафоры 1 | Головин И.Г. |
| 43 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('2');  putchar('3');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('1');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  23321  23231  22313  22133  22331  12323  22231  12233 | **Невозможные варианты вывода:**  **1) 23321**  **4) 22133**  **7)22231**  Семафоры 1 | Семенов А.Н. |
| 44 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 2);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  abbac  ababc  caabb  aacbb  aabbc  cabab | **Невозможные варианты вывода:**  **1) abbac**  **3) caabb**  **4) aacbb**  Семафоры 1 | Семенов А.Н. |
| 45 | **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **Вычеркните варианты вывода, невозможные при выполнении фрагментов программ тремя параллельными процессами (A,B,C):**  semId – идентификатор массива семафоров, состоящего из 1 семафора. Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 1);  Данный фрагмент выполняется двумя параллельными процессами (**A и B**):  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('a');  putchar('b');  op.sem\_op = 1;  semop(semId, &op, 1);  Данный фрагмент выполняется одним параллельным процессом С:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -1;  semop(semId, &op, 1);  putchar('c');  op.sem\_op = 1;  semop(semId, &op, 1);  **Варианты вывода:**  abbac  ababc  aabcb  aacbb  aabbc  cabab | **Невозможные варианты вывода:**  **1) abbac**  **3) aabcb**  **4) aacbb**  **5) aabbc**  Семафоры 1 | Семенов А.Н. |

**Задача 4, очереди сообщений**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | 132 | Базовый |
| 2 | Что будет выведено на экран в результате работы фрагмента программы? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct {  long type;  char data[1];  } msg;  ……..  msg.type = 1; msg.data[0] = 'a'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = 'b'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = 'c'; msgsnd(msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = 'd'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);  …… | adb | Казачук М.А. |
| 3 | Что будет выведено на экран в результате работы фрагмента программы? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct {  long type;  char data[1];  } msg;  ……..  msg.type = 1; msg.data[0] = 'a'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = 'b'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = 'c'; msgsnd(msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = 'd'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0); putchar(msg.data[0]);  …… | bca | Казачук М.А. |
| 4 | Что будет выведено на экран в результате работы фрагмента программы? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  ……..  msg.type = 1; msg.data[0] = '1'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = '2'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = '3'; msgsnd(msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = '4'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0); putchar(msg.data[0]);  …… | 214 | Казачук М.А. |
| 5 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **132**  Очереди сообщений | Кузина Л.Н. |
| 6 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 2;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]); | **cbad**  Очереди сообщений | Кузина Л.Н. |
| 7 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, -2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | 1**423F**  Очереди сообщений | Кузина Л.Н. |
| 8 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **msgId – идентификатор существующей пустой очереди сообщений.**  **struct** {  **long** type;  **char** data[1];  } msg;  msg.type = 2; msg.data[0] = '1'; msgsnd (msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = '2'; msgsnd(msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = '3'; msgsnd(msgId, &msg, 1, 0);  msg.type = 3; msg.data[0] = '4'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);    putchar ('7'); | Ответ: 43127 | Волкова И.А. |
| 9 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **msgId – идентификатор существующей пустой очереди сообщений.**  **struct** {  **long** type;  **char** data[1];  } msg;  msg.type = 1; msg.data[0] = '1'; msgsnd (msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = '2'; msgsnd(msgId, &msg, 1, 0);  msg.type = 3; msg.data[0] = '3'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = '4'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);    putchar ('7'); | Ответ: 12347 | Волкова И.А. |
| 10 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **msgId – идентификатор существующей пустой очереди сообщений.**  **struct** {  **long** type;  **char** data[1];  } msg;  msg.type = 2; msg.data[0] = '1'; msgsnd (msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = '2'; msgsnd(msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = '3'; msgsnd(msgId, &msg, 1, 0);  msg.type = 3; msg.data[0] = '4'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);    putchar ('7'); | Ответ: 21 | Волкова И.А. |
| 11 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'z';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'v';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('a'); | **zcx** | Корныхин Е.В. |
| 12 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'f';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'g';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'h';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'j';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  putchar('v'); | **fhg** | Корныхин Е.В. |
| 13 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  putchar('6'); | **214** | Корныхин Е.В. |
| 14 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 10;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 15;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 15;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, -10, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 15, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]); | **cabd**  Очереди сообщений | Тюляева В.В. |
| 15 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 4;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, -3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, -2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]); | **dcba**  Очереди сообщений | Тюляева В.В. |
| 16 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('X'); | **acb**  Очереди сообщений | Гомзин А.Г. |
| 17 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **adb**  Очереди сообщений | Гомзин А.Г. |
| 18 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '0';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('Z'); | **312**  Очереди сообщений | Гомзин А.Г. |
| 19 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '9';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '5';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '7';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '0';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **975** | Никольский И.М. |
| 20 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '8';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '9';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '7';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '5';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('Q'); | **879** | Никольский И.М. |
| 21 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'q';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'w';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'e';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'r';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('V'); | **qew** | Никольский И.М. |
| 22 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  **struct**  {  **long** type;  **char** data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]); | 132  Очереди сообщений | Вылиток А.А. |
| 23 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  **struct**  {  **long** type;  **char** data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]); | 13d  Очереди сообщений | Вылиток А.А. |
| 24 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  **struct**  {  **long** type;  **char** data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'z';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'y';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'u';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]); | zxy  Очереди сообщений | Вылиток А.А. |
| 25 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msg.type = 4;  msg.data[0] = ‘e';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **acb**  Очереди сообщений | Глазкова В.В. |
| 26 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **132F**  Очереди сообщений | Глазкова В.В. |
| 27 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **312F**  Очереди сообщений | Глазкова В.В. |
| 28 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'y';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'z';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'w';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | 4. xzy | Дряженков А.А. |
| 29 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'y';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'z';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'w';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | 12. zxy | Дряженков А.А. |
| 30 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'y';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'z';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'w';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  putchar('F'); | 20. zxw | Дряженков А.А. |
| 31 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('A'); | **312**  Очереди сообщений | Полякова И.Н. |
| 32 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'r';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'y';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'm';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('c'); | **rxy**  Очереди сообщений | Полякова И.Н. |
| 33 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 1;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('4'); | **21**  Очереди сообщений | Полякова И.Н. |
| 34 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 2;  msg.data[0] = 'x';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'y';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'z';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 't';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]); | **xzy** | Жуков К.А. |
| 35 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 1;  msg.data[0] = 'm';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'k';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'p';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 's';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  **putchar(msg.data[0]);** | **kms** | Жуков К.А. |
| 36 | **Что будет выведено на экран в результате работы фрагмента программы? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.** msgId – идентификатор существующей пустой очереди сообщений.  struct {  long type;  char data[1]; } msg;  ……**..**  msg.type = 1;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'e';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]); | **cae** | Жуков К.А. |
| 37 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 2;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **132**  Очереди сообщений | Шестимеров А.А. |
| 38 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **231**  Очереди сообщений | Шестимеров А.А. |
| 39 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **132**  Очереди сообщений | Шестимеров А.А. |
| 40 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar(‘F’); | **1342F**  Очереди сообщений | Сучков Е.П. |
| 41 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar(‘F’); | **123** | Сучков Е.П. |
| 42 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar(‘F’); | **312** | Сучков Е.П. |
| 43 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 1; msg.data[0] = 'a'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = 'b'; msgsnd(msgId, &msg, 1, 0);  msg.type = 2; msg.data[0] = 'c'; msgsnd(msgId, &msg, 1, 0);  msg.type = 1; msg.data[0] = 'd'; msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0); putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **bca**  Очереди сообщений | Головин И.Г. |
| 44 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  putchar('F'); | **abd**  Очереди сообщений | Головин И.Г. |
| 45 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 1;  msg.data[0] = 'a';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = 'b';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = 'c';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = 'd';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  putchar('F'); | **ab**  Очереди сообщений | Головин И.Г. |
| 46 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 1, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **1324**  Очереди сообщений | Семенов А.Н. |
| 47 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  putchar('F'); | **124**  Очереди сообщений | Семенов А.Н. |
| 48 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  msgId – идентификатор существующей пустой очереди сообщений.  struct  {  long type;  char data[1];  } msg;  …  msg.type = 3;  msg.data[0] = '1';  msgsnd(msgId, &msg, 1, 0);  msg.type = 2;  msg.data[0] = '2';  msgsnd(msgId, &msg, 1, 0);  msg.type = 1;  msg.data[0] = '3';  msgsnd(msgId, &msg, 1, 0);  msg.type = 3;  msg.data[0] = '4';  msgsnd(msgId, &msg, 1, 0);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 3, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 2, 0);  putchar(msg.data[0]);  msgrcv(msgId, &msg, 1, 0, 0);  putchar(msg.data[0]);  putchar('F'); | **1423F**  Очереди сообщений | Семенов А.Н. |

**Задача 5, semaphores, type 2**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  bababa  babbaa  bbabaa  bbaaba | Базовый |
| 2 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flg = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "1", 1);  write(1, "2", 1);  op.sem\_op = 2;  semop(semId, &op, 1); | Варианты:  121212 | Казачук М.А. |
| 3 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op; op.sem\_num = 0; op.sem\_flg = 0;  op.sem\_op = -2; semop(semId, &op, 1);  write(1, "a", 1); write(1, «b", 1);  op.sem\_op = 2; semop(semId, &op, 1); | Варианты:  ababab  abaabb  aababb  aabbab | Казачук М.А. |
| 4 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flg = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  putchar('e');  putchar('f');  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  efefef  efeeff  eefeff  eeffef | Казачук М.А. |
| 5 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bababa**  **babbaa**  **bbabaa**  **bbaaba**  Семафоры 2 | Кузина Л.Н. |
| 6 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  write(1, "x", 1);  write(1, "y", 1);  op.sem\_op = 2;  **semop(semId, &op, 1);** | Варианты:  **xyxyxy**  **xxyyxy**  **xxyxyy**  **xyxxyy**  Семафоры 3 | Кузина Л.Н. |
| 7 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop (semId, &op, 1);  write (1, "1", 1);  write (1, "5", 1);  op.sem\_op = 3;  semop (semId, &op, 1); | 151515 или  115515 или  151155 или  115155 | Волкова И.А. |
| 8 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop (semId, &op, 1);  write (1, "a", 1);  write (1, "c", 1);  op.sem\_op = 2;  semop (semId, &op, 1); | aaaccc или  acacac или  aacacc или  acaacc или  aaccac или | Волкова И.А. |
| 9 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации. Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 4);**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop (semId, &op, 1);  write (1, "a", 1);  write (1, "b", 1);  op.sem\_op = 2;  semop (semId, &op, 1); | aabbab или  ababab или  abaabb или  aababb или | Волкова И.А. |
| 10 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора.**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "с", 1);  write(1, "d", 1);  op.sem\_op = 4;  semop(semId, &op, 1); | Варианты:  **cdcdcd**  **cdccdd** | Корныхин Е.В. |
| 11 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора.**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 9);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  write(1, "x", 1);  write(1, "y", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **xyxyxy**  **xyxxyy**  **xxyxyy**  **xxyyxy** | Корныхин Е.В. |
| 12 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора.**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  write(1, "q", 1);  write(1, "w", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **qwqwqw**  **qwqqww**  **qqwqww**  **qqwwqw** | Корныхин Е.В. |
| 13 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 2 семафоров**  **Массив проинициализирован нулями.**  **Фрагмент программы выполняется 2-мя параллельными процессами:**  **1 процесс:**  struct sembuf p0;  p0.sem\_num = 0;  p0.sem\_flag = 0;  struct sembuf v1;  v1.sem\_num = 1;  v1.sem\_flag = 0;  write(1, "a", 1);  v1.sem\_op = 1;  semop(semId, &v1, 1);  p0.sem\_op = -1;  semop(semId, &p0, 1);  write(1, "b", 1);  **2 процесс:**  struct sembuf p1;  p1.sem\_num = 1;  p1.sem\_flag = 0;  struct sembuf v0;  v0.sem\_num = 0;  v0.sem\_flag = 0;  p1.sem\_op =-1;  semop(semId, &p1, 1);  write(1, "c", 1);  v0.sem\_op=1;  semop(semId, &v0, 1);  write(1, "d", 1); | Варианты:  **acbd или acdb**  Семафоры 2 | Тюляева В.В. |
| 14 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафорa**  **Массив проинициализирован нулями.**  **Фрагмент программы выполняется 2-мя параллельными процессами:**  **1 процесс:**  struct sembuf p;  p.sem\_num = 0;  p.sem\_flag = 0;  write(1, "a", 1);  p.sem\_op = 2;  semop(semId, &p, 1);  p.sem\_op = 0;  semop(semId, &p, 1);  write(1, "b", 1);  **2 процесс:**  struct sembuf p;  p.sem\_num = 0;  p.sem\_flag = 0;  p.sem\_op =-1;  semop(semId, &p, 1);  write(1, "c", 1);  semop(semId, &p, 1);  write(1, "d", 1); | Варианты:  **acbd или acdb**  Семафоры 2 | Тюляева В.В. |
| 15 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "a", 1);  write(1, "b", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **ababab**  **abaabb**  **aababb**  **aabbab**  Семафоры 2 | Гомзин А.Г. |
| 16 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "1", 1);  write(1, "2", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **121212**  **121122**  **112122**  **112212**  Семафоры 2 | Гомзин А.Г. |
| 17 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "bz\_", 3);  write(1, "ax\_", 3);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bz\_ax\_bz\_ax\_bz\_ax\_**  **bz\_ax\_bz\_bz\_ax\_ax\_**  **bz\_bz\_ax\_bz\_ax\_ax\_**  **bz\_bz\_ax\_ax\_bz\_ax\_**  Семафоры 2 | Гомзин А.Г. |
| 18 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "r", 1);  write(1, "t", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **rtrtrt**  **rtrrtt**  **rrtrtt**  **rrttrt** | Никольский И.М. |
| 19 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "v", 1);  write(1, "u", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **vuvuvu**  **vuvvuu**  **vvuvuu**  **vvuuvu** | Никольский И.М. |
| 20 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "z", 1);  write(1, "y", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **zyzyzy**  **zyzzyy**  **zzyzyy**  **zzyyzy** | Никольский И.М. |
| 21 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора.  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "c", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  cacaca  caccaa  ccacaa  ccaaca  Семафоры, 2 | Вылиток А.А. |
| 22 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора.  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "1", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  1a1a1a  1a11aa  11a1aa  11aa1a  Семафоры, 2 | Вылиток А.А. |
| 23 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора.  Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 8);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "x", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  xaxaxa  xaxxaa  xxaxaa  xxaaxa  Семафоры, 2 | Вылиток А.А. |
| 24 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 4;  semop(semId, &op, 1); | Варианты:  **bababa**  Семафоры 2 | Глазкова В.В. |
| 25 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  write(1, "1", 1);  write(1, "2", 1);  op.sem\_op = 2;  semop(semId, &op, 1); | Варианты:  **121212**  **121122**  **112122**  **112212**  Семафоры 2 | Глазкова В.В. |
| 26 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bababa**  Семафоры 2 | Глазкова В.В. |
| 27 | Что будет выведено на экран, если PID изначально запущенного процесса равен **156**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int main() {**  **int pid;**  **if ( ! fork() ) printf ( "PPID = %d \n", getppid() );**  **else** exit(0);  **return** 0;  **}** | 5. PPID=156 или PPID=1 | Дряженков А.А. |
| 28 | Что будет выведено на экран, если PID изначально запущенного процесса равен **30123**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int main() {**  **int pid;**  **if ( fork() )**  exit(0);  **else**  **printf ( "PPID = %d \n", getppid() );**  **return** 0;  **}** | 13. PPID=30123 или PPID=1 | Дряженков А.А. |
| 29 | Что будет выведено на экран, если PID изначально запущенного процесса равен **12330**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int main() {**  **int pid;**  **if ( fork() )**  exit(0);  **else**  **printf ( "PPID = %d \n", getppid() );**  **return** 0;  **}** | 21. PPID=12330 или PPID=1 | Дряженков А.А. |
| 30 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "a", 1);  write(1, "b", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **ababab**  **abaabb**  **aababb**  **aabbab**  Семафоры 2 | Полякова И.Н. |
| 31 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "4", 1);  write(1, "y", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **4y4y4y**  **4y44yy**  **44y4yy**  **44yy4y**  Семафоры 2 | Полякова И.Н. |
| 32 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "xy", 2);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **xyaxyaxya**  **xyaxyxyaa**  **xyxyaxyaa**  **xyxyaaxya**  Семафоры 2 | Полякова И.Н. |
| 33 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  printf(“5”);  printf(“4”);  op.sem\_op = 2;  semop(semId, &op, 1); | **545454**  **545544**  **554544**  **554454** | Жуков К.А. |
| 34 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 9);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  write(1, "x", 1);  write(1, "y", 1);  op.sem\_op = 4;  semop(semId, &op, 1); | Варианты:  **xyxyxy**  **xyxxyy**  **xxyxyy**  **xxyyxy** | Жуков К.А. |
| 35 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId** – **идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op; op.sem\_num = 0; op.sem\_flag = 0; op.sem\_op = -3; semop(semId, &op, 1); putchar('e'); putchar('f'); op.sem\_op = 3; semop(semId, &op, 1); | efefef либо efeeff либо eefeff либо eeffef | Жуков К.А. |
| 36 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 1;  semop(semId, &op, 1); | Варианты:  **bababa**  **bbaaba**  Семафоры 2 | Шестимеров А.А. |
| 37 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 6);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "a", 1);  write(1, "b", 1);  op.sem\_op = 2;  semop(semId, &op, 1); | Варианты:  **bababa**  **bbaaba**  Семафоры 2 | Шестимеров А.А. |
| 38 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 3);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 1;  semop(semId, &op, 1); | Варианты:  **baba**  Семафоры 2 | Шестимеров А.А. |
| 39 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 5);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **semctl(semId, 0, SETVAL, 5);**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  write(1, "a", 1);  write(1, "b", 1);  op.sem\_op = 2;  semop(semId, &op, 1); | Варианты:  **ababab**  **abaabb**  **aabbab**  **aababb**  Семафоры 2 | Сучков Е.П. |
| 40 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **semctl(semId, 0, SETVAL, 7);**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 6;  semop(semId, &op, 1); | bababa, babbaa | Сучков Е.П. |
| 41 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **semctl(semId, 0, SETVAL, 7);**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -4;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 4;  semop(semId, &op, 1); | bababa | Сучков Е.П. |
| 42 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 6);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bababa**  **babbaa**  **bbabaa**  **bbaaba**  Семафоры 2 | Головин И.Г. |
| 43 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 9);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bababa**  **babbaa**  **bbabaa**  **bbaaba**  **bbbaaa**  Семафоры 2 | Головин И.Г. |
| 44 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 1;  semop(semId, &op, 1); | Варианты:  **bababa**  **bbaaba**  Семафоры 2 | Головин И.Г. |
| 45 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  **char q =** "b";  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, q++, 1);  write(1, q++, 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bcbcbc**  **bcbbcc**  **bbcbcc**  **bbccbc**  **В задании используется q++ в вызове функции, что undefined behaviour, на**  **что студент при ответе должен указать.**  Семафоры 2 | Семенов А.Н. |
| 46 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 7);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -3;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 3;  semop(semId, &op, 1); | Варианты:  **bababa**  **babbaa**  **bbabaa**  **bbaaba**  Семафоры 2 | Семенов А.Н. |
| 47 | **Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **semId – идентификатор массива семафоров, состоящего из 1 семафора**  **Массив проинициализирован с помощью вызова: semctl(semId, 0, SETVAL, 6);**  **Фрагмент программы выполняется 3-мя параллельными процессами:**  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -2;  semop(semId, &op, 1);  write(1, "b", 1);  write(1, "a", 1);  op.sem\_op = 2;  semop(semId, &op, 1); | Варианты:  **bababa**  **babbaa**  **bbabaa**  **bbaaba**  **bbbaaa**  Семафоры 2 | Семенов А.Н. |

**Задача 6, fork + чтение (разделяемого) файла**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | Ответ: aad либо cad  Комментарий для проверяющего:  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый. | Базовый |
| 2 | Каким будет содержание файла file после каждого из вызовов write?  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  fd1 = open(file, O\_RDWR | O\_CREAT | O\_TRUNC, S\_IRUSR | S\_IWUSR);  fd2 = dup(fd1);  fd3 = open(file, O\_RDWR);  write(fd1, "Hello,", 6); //1  write(fd2, "world", 6); //2  lseek(fd2, 0, SEEK\_SET);  if(!fork)  {  write(fd1, "HELLO,", 6); //3  write(fd3, "123456", 6); //4  \_exit(0);  }  wait(NULL);  write(fd2, "ByeBye", 6); //5 | 1 – Hello,  2 – Hello,world  3 – HELLO,world  4 – 123456world  5 – 123456ByeBye | Казачук М.А. |
| 3 | Содержимое файла “1.txt” – строка «abcde».  Возможны ли варианты вывода:  cad и abc?  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | Возможны: aad либо cad  Невозможны: abc  Комментарий для проверяющего:  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый. | Казачук М.А. |
| 4 | Содержимое файла “1.txt” – строка «HelloWorld». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, -6, SEEK\_END);  wait(NULL);  read(fd3, &c, 1); write(1, &c, 1);  read(fd2, &c, 1); write(1, &c, 1);  read(fd, &c, 1); write(1, &c, 1);  }  else  {  read(fd, &c, 1); write(1, &c, 1);  }  return 0;  } | oWHo или HoHW  Комментарий для проверяющего:  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый. | Казачук М.А. |
| 5 | Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aad либо cad**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Кузина Л.Н. |
| 6 | Содержимое файла “1.txt” – строка «123456». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = '0';  int fd1,fd2,fd3;  fd1 = open("1.txt", O\_RDONLY);  if(fork())  {  fd3 = dup(fd1);  lseek(fd1, 2, SEEK\_CUR);  wait(NULL);  read(fd3, &c, 1); write(1, &c, 1);  read(fd1, &c, 1); write(1, &c, 1);  }  else  {  read(fd1, &c, 1); write(1, &c, 1);  fd2 = open("1.txt", O\_RDONLY);  read(fd2, &c, 1); write(1, &c, 1);  }  return 0;  } | **Ответ: 3145 либо 1145**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Кузина Л.Н. |
| 7 | **Содержимое файла “1.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = '0';  int fd1,fd2,fd3;  fd1 = open("1.txt", O\_RDONLY);  if(fork())  {  fd3 = dup(fd1);  lseek(fd3, 1, SEEK\_CUR);  wait(NULL);  read(fd1, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  {  read(fd1, &c, 1); write(1, &c, 1);  fd2 = open("1.txt", O\_RDONLY);  read(fd2, &c, 1); write(1, &c, 1);  }  return 0;  } | **Ответ: 2134 либо 1134**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Кузина Л.Н. |
| 8 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Int main(){  **char** c = 'a';  **int** fd;    fd = open ("1.txt", O\_RDONLY);  **if** (fork())  {  **int** fd2 = open ("1.txt", O\_RDONLY);  **int** fdd = dup (fd);  lseek (fd, 1, SEEK\_CUR);  wait (NULL);  read (fd2, &c, 1);  write(1, &c, 1);  read (fdd, &c, 1);  write(1, &c, 1);  }  else {  read (fd, &c, 1);  write (1, &c, 1);  }  **return** 0;  } | **Ответ: aac либо bac** | Волкова И.А. |
| 9 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Int main{  **char** c = 'a';  **int** fd;    fd = open ("1.txt", O\_RDONLY);  **if** (fork())  {  **int** fd2 = open ("1.txt", O\_RDONLY);  **int** fdd = dup (fd);  lseek (fd, 3, SEEK\_SET);  read (fd2, &c, 1);  write(1, &c, 1);  wait (NULL);  read (fdd, &c, 1);  write(1, &c, 1);  }  else {  read (fd, &c, 1);  write (1, &c, 1);  }  **return** 0;  } | **Ответ: aad либо dae либо ade** | Волкова И.А. |
| 10 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  Int main{  **char** c = 'a';  **int** fd;    fd = open ("1.txt", O\_RDONLY);  **if** (fork() == 0)  {  **int** fd2 = open ("1.txt", O\_RDONLY);  **int** fdd = dup (fd);  lseek (fd, 3, SEEK\_CUR);  read (fd2, &c, 1);  write(1, &c, 1);  read (fdd, &c, 1);  write(1, &c, 1);  }  read (fd, &c, 1);  write (1, &c, 1);    **return** 0;  } | Ответ: **aae либо dae**  **либо ade** | Волкова И.А. |
| 11 | **Содержимое файла “input.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main(void)  {  char c = 'a';  int fd = open("input.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("input.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 3, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **115 или 415** | Корныхин Е.В. |
| 12 | **Содержимое файла “input.txt” – строка «zxcv». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main(void)  {  char c = 'a';  int fd2, fd3, fd = open("input.txt", O\_RDONLY);  if(fork() == 0)  { read(fd, &c, 1); write(1, &c, 1); exit(0); }  else  {  fd2 = open("input.txt", O\_RDONLY);  fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  }  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  return 0;  } | **zzv или czv** | Корныхин Е.В. |
| 13 | **Содержимое файла “input” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main(void)  {  int fd = open("input", O\_RDONLY);  char c = 'a';  if(fork())  {  int fd2 = dup(fd);  int fd3 = open("input", O\_RDONLY);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **141 или 341** | Корныхин Е.В. |
| 14 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = dup(fd);  lseek(fd2, 2, SEEK\_CUR);  wait(NULL);  read(fd, &c, 1); write(1, &c, 1);  read(fd2, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: ade либо cde**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Тюляева В.В. |
| 15 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2=dup(fd);  lseek(fd2, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd, &c, 1); write(1, &c, 1);  }  else  { int fd2 = open("1.txt", O\_RDONLY);  read(fd2, &c, 1); write(1, &c, 1);  }  return 0;  } | **Ответ: acd**  Форк+чтение (разделяемого) файла | Тюляева В.В. |
| 16 | **Содержимое файла “1.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd2);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: 112 либо 312**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Гомзин А.Г. |
| 17 | **Содержимое файла “1.txt” – строка «close». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd3, &c, 1); write(1, &c, 1);  read(fd2, &c, 1); write(1, &c, 1);    }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: csc либо osc**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Гомзин А.Г. |
| 18 | **Содержимое файла “1.txt” – строка «touch». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: ttc либо utc**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла |  |
| 19 | **Содержимое файла “dat.txt” – строка «kgptr». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'k';  int fd;    fd = open("dat.txt", O\_RDONLY);  if(fork())  {  int fd3 = dup(fd);  int fd2 = open("dat.txt", O\_RDONLY);    lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: kkt либо pkt** | Никольский И.М. |
| 20 | **Содержимое файла “dat.txt” – строка «ftypj». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'f';  int fd;    fd = open("dat.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("dat.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: ffp либо yfp** | Никольский И.М. |
| 21 | **Содержимое файла “ft.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = '1';  int fd;    fd = open("ft.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("ft.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: 114 либо 314** | Никольский И.М. |
| 22 | Содержимое файла “1.txt” – строка «a1c2e». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  **char** c = 'x';  **int** fd;    fd = open("1.txt", O\_RDONLY);  **if**(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  **return** 0;  }    read(fd, &c, 1); write(1, &c, 1);  **return** 0;  } | Ответ: aa2 либо ca2  Комментарий для проверяющего:  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Вылиток А.А. |
| 23 | Содержимое файла “1.txt” – строка «b1c2e». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  **char** c = 'y';  **int** fd;    fd = open("1.txt", O\_RDONLY);  **if**(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  **return** 0;  }    read(fd, &c, 1); write(1, &c, 1);  **return** 0;  } | Ответ: bb2 либо cb2  Комментарий для проверяющего:  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Вылиток А.А. |
| 24 | Содержимое файла “1.txt” – строка «1x2y3». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int** main()  {  **char** c = '1';  **int** fd;    fd = open("1.txt", O\_RDONLY);  **if**(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  **return** 0;  }    read(fd, &c, 1); write(1, &c, 1);  **return** 0;  } | Ответ: 11y либо 21y  Комментарий для проверяющего:  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Вылиток А.А. |
| 25 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); }  return 0;  } | **Ответ: ad**  Форк+чтение (разделяемого) файла | Глазкова В.В. |
| 26 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: ad либо cd**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Глазкова В.В. |
| 27 | **Содержимое файла “1.txt” – строка «edcba». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'e';  int fd;    fd = open("1.txt", O\_RDONLY);  if(!fork())  {  read(fd, &c, 1); write(1, &c, 1);  }  else  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  return 0;  } | **Ответ: eeb либо ceb**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Глазкова В.В. |
| 28 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова:  semctl(semId, 0, SETVAL, 11);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -5;  semop(semId, &op, 1);  write(1, "1", 1);  write(1, "2", 1);  op.sem\_op = 5;  semop(semId, &op, 1); | 6. 121212  121122  112122  112212 | Дряженков А.А. |
| 29 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова:  semctl(semId, 0, SETVAL, 11);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -5;  semop(semId, &op, 1);  write(1, "5", 1);  write(1, "4", 1);  op.sem\_op = 5;  semop(semId, &op, 1); | 14. 545454  545544  554544  554454 | Дряженков А.А. |
| 30 | Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  semId – идентификатор массива семафоров, состоящего из 1 семафора  Массив проинициализирован с помощью вызова:  semctl(semId, 0, SETVAL, 11);  Фрагмент программы выполняется 3-мя параллельными процессами:  struct sembuf op;  op.sem\_num = 0;  op.sem\_flag = 0;  op.sem\_op = -5;  semop(semId, &op, 1);  write(1, "4", 1);  write(1, "5", 1);  op.sem\_op = 5;  semop(semId, &op, 1); | 22. 454545  454455  445455  445545 | Дряженков А.А. |
| 31 | **Содержимое файла “1.txt” – строка «bdcea». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c;  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: bbe либо cbe**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Полякова И.Н. |
| 32 | **Содержимое файла “tt.txt” – строка «41286». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("tt.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("tt.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: 448 либо 248**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Полякова И.Н. |
| 33 | **Содержимое файла “1.txt” – строка «xyzxyz». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char t = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &t, 1); write(1, &t, 1);  read(fd3, &t, 1); write(1, &t, 1);  }  else  { read(fd, &t, 1); write(1, &t, 1); }  return 0;  } | **Ответ: xxx либо zxx**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Полякова И.Н. |
| 34 | **Содержимое файла “2.txt” – строка «56789». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = '3';  int fd;    fd = open("2.txt", O\_RDONLY);  if(fork())  {  int fd3 = dup(fd);  int fd2 = open("2.txt", O\_RDONLY);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); putchar(c);  read(fd3, &c, 1); putchar(c);  }  else  { read(fd, &c, 1); putchar(c); }  return 0;  } | **Ответ: 558 либо 758** | Жуков К.А. |
| 35 | **Содержимое файла “3.txt” – строка «uwxyz». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'x';  int fd;    fd = open("3.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("3.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); } | **Ответ: uuy либо xuy** | Жуков К.А. |
| 36 | **Содержимое файла “4.txt” – строка «54321». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = '1’;  int fd;    fd = open("4.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("4.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); printf(“%c”, c);  read(fd3, &c, 1); printf(“%c”, c);  }  else  { read(fd, &c, 1); printf(“%c”, c); } | **Ответ: 552 либо 352** | Жуков К.А. |
| 37 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 3, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aae либо dae**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Шестимеров А.А. |
| 38 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;  lseek(fd, 1, SEEK\_CUR);  fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: bad либо cad**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Шестимеров А.А. |
| 39 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, -2, SEEK\_END);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aa\0 либо ea\0**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Шестимеров А.А. |
| 40 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd3, 3, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aae или dae**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Сучков Е.П. |
| 41 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  lseek(fd, 2, SEEK\_CUR);  int fd3 = dup(fd);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **cad или aad** | Сучков Е.П. |
| 42 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd2);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **cab или aab** | Сучков Е.П. |
| 43 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, -2, SEEK\_END);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aae либо dae**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Головин И.Г. |
| 44 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'x', d=’x’;  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  read(fd3, &d, 1);  wait(NULL);  read(fd2, &c, 1);write(1, &c, 1);  write(1, &d, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aae либо dac либо cad**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Вторые «гонки» также возникают между read(fd[0]…) в отце и сыне, отсюда третий вариант  Форк+чтение (разделяемого) файла | Головин И.Г. |
| 45 | **Содержимое файла “1.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char x = ‘1’;  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  wait(NULL);  read(fd2, &x, sizeof x); write(1, &x, sizeof x);  read(fd3, &x, sizeof x); write(1, &x, sizeof x);  }  else  {  lseek(fd, 2, SEEK\_CUR);  read(fd, &x, sizeof x); write(1, &x, sizeof x);    }  return 0;  } | **Ответ: 314**  **Комментарий для проверяющего:**  Гонок нет. Вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Головин И.Г. |
| 46 | **Содержимое файла “1.txt” – строка «abcab». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: aaa либо caa**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Семенов А.Н. |
| 47 | **Содержимое файла “1.txt” – строка «abcde». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1);  read(fd, &c, 1); write(1, &c, 1);  }  return 0;  } | **Ответ: abae либо cdae либо adae**  **Комментарий для проверяющего:**  Три варианта порождаются за счет «гонок» между lseek и двумя read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Семенов А.Н. |
| 48 | **Содержимое файла “1.txt” – строка «exam!». Что будет выведено на экран? Если возможны несколько вариантов – привести все.**  **Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.**  **Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  int main()  {  char c = 'a';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | **Ответ: eem либо aem**  **Комментарий для проверяющего:**  Два варианта порождаются за счет «гонок» между lseek и read в «сыне». Сам вывод определяется тем, что при «наследовании» и дублировании файлового дескриптора файловый указатель является общим, а при open создается новый.  Форк+чтение (разделяемого) файла | Семенов А.Н. |

**Задача 7, Базовое: создание/завершение процесса**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Что будет выведено на экран, если PID изначально запущенного процесса равен 1277? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  main() {  int pid;  if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )  else exit();  } | Ответ: PPID=1277 или PPID=1  - в зависимости от того, как сработает планировщик | Базовый |
| 2 | Пусть процесс с PID **D** породил два сыновьих процесса с PID-ами **C** и **A**:  int main(int argc, char \*\*argv) //PID = D  {  if (fork() == 0){ //PID = C  printf ("%d %d\n", getppid(), getpid());  exit(0);  }  if (fork() == 0){ //PID = A  printf ("%d\n", getpid());  exit(0);  }  return 0;  }  Считаем, что printf работает атомарно и обращения ко всем системным вызовам успешно отрабатывают. Перечислить **все** возможные комбинации значений, которые могут быть выведены на стандартное устройство вывода в результате выполнения данной программы. | **D C**  **A**  либо  **1 C**  **A**  либо  **A**  **D C**  либо  **A**  **1 C** | Казачук М.А. |
| 3 | Пусть процесс с PID **A** породил сыновий процесс с PID **B**:  int main(int argc, char \*\*argv) //PID = A  {  int n = 42;  if (fork() == 0){ //PID = B  printf ("%d %d %d\n", n, getppid(), getpid());  n = 8;  exit(0);  }  n = 10;  printf ("%d %d\n", n, getpid());  return 0;  }  Считаем, что printf работает атомарно, без буферизации, и обращения ко всем системным вызовам успешно отрабатывают. Перечислить **все** возможные комбинации значений, которые могут быть выведены на стандартное устройство вывода в результате выполнения данной программы. | **42 A B**  **10 A**  либо  **10 A**  **42 A B**  либо  **10 A**  **42 1 B** | Казачук М.А. |
| 4 | Пусть процесс с PID 4123 породил два сыновних процесса с PID-ами 4124и 4125:  int main(int argc, char \*\*argv) //PID = 4123  {  if (fork() == 0){ //PID = 4124  printf ("%d \n", getpid());  exit(0);  }  wait(NULL);  if (fork() == 0){ //PID = 4125  printf ("%d %d \n", getpid(), getppid());  exit(0);  }  return 0;  }  Считаем, что printf работает атомарно и обращения ко всем системным вызовам успешно отрабатывают. Перечислить **все** возможные комбинации значений, которые могут быть выведены на стандартное устройство вывода в результате выполнения данной программы. | **4124**  **4125 4123**  либо  **4124**  **4125 1** | Казачук М.А. |
| 5 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )**  **else** exit();  **}** | Ответ: PPID=1277 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Кузина Л.Н. |
| 6 | **Что будет выведено на экран, если PID изначально запущенного процесса равен 1277? Если возможны несколько вариантов – обосновать и привести все варианты.** Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int** pid**;**  **if** ( fork() == 0 )  printf ( "PPID = %d \n", getppid() );  **else** {  wait (NULL);  exit (0);  }  **}** | Ответ: 1277 | Волкова И.А. |
| 7 | **Что будет выведено на экран, если PID изначально запущенного процесса равен 1555? Если возможны несколько вариантов – обосновать и привести все варианты.** Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int** pid**;**  **if** (fork() ) **{**  **if** (fork() == 0)  printf ( "%d \n", getppid() );  }  **else**  exit (0);  **}** | Ответ: 1555 или 1 | Волкова И.А. |
| 8 | **Что будет выведено на экран, если PID изначально запущенного процесса равен 777? Если возможны несколько вариантов – обосновать и привести все варианты.** Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int** pid**;**  **if** ( fork() )  **if** (fork())  **if** (fork() == 0 )  printf ( "%d \n", getppid() )  exit (0);  **}** | Ответ: 777 или 1 | Волкова И.А. |
| 9 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1234**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int main(void) {**  **if (fork() == 0) printf ( "%d \n", getppid());**  **}** | 1234 или 1 | Корныхин Е.В. |
| 10 | Что будет выведено на экран, если PID изначально запущенного процесса равен **3344**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int main(void) {**  **if (fork())** exit(0);  **printf ("%d\n", getppid());**  **}** | 3344 или 1 | Корныхин Е.В. |
| 11 | Что будет выведено на экран, если PID изначально запущенного процесса равен **5555**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **int main(void) {**  **int pid;**  **if ((pid = fork()) == 0 ) {  printf("P = %d\n", getppid());  }**  **}** | P = 5555 или P = 1 | Корныхин Е.В. |
| 12 | Что будет выведено на экран, если PID изначально запущенного процесса равен **2022**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( (pid = fork()) == 0 )**  **printf ( "PPID = %d \n", getppid() );**  **else** exit();  **}** | Ответ: PPID=2022 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Гомзин А.Г. |
| 13 | Что будет выведено на экран, если PID изначально запущенного процесса равен **2123**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() != 0 ) exit();**  **else** **printf ( "PPID = %d \n", getppid()**;  **}** | Ответ: PPID=2123 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Гомзин А.Г. |
| 14 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1024**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "P = %d \n", getppid() );**  **else** exit();  **}** | Ответ: P=1024 или P=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Гомзин А.Г. |
| 15 | **Что будет выведено на экран, если PID изначально запущенного процесса равен 3145? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else exit(0);**  **}** | Ответ: PPID=3145 или PPID=1 | Никольский И.М. |
| 16 | **Что будет выведено на экран, если PID изначально запущенного процесса равен 8425? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.**  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else exit(0);**  **}** | Ответ: PPID=8425 или PPID=1 | Никольский И.М. |
| 17 | Что будет выведено на экран, если PID изначально запущенного процесса равен **8156**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else** exit(0);  **}** | Ответ: PPID=8156 или PPID=1 | Никольский И.М. |
| 18 | Что будет выведено на экран, если PID изначально запущенного процесса равен 1271? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  main() {  **int** pid;  **if** ( fork() == 0 )  printf ( "PPID = %d \n", getppid() )  **else** **return** 0;  } | Ответ: PPID=1271 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Вылиток А.А. |
| 19 | Что будет выведено на экран, если PID изначально запущенного процесса равен 1071? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  main() {  **int** pid;  **if** ( fork() == 0 )  printf ( "PPID = %d \n", getppid() )  **else** **return** 0;  } | Ответ: PPID=1071 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Вылиток А.А. |
| 20 | Что будет выведено на экран, если PID изначально запущенного процесса равен 1572? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  main() {  **int** pid;  **if** ( fork() == 0 )  printf ( "PPID = %d \n", getppid() )  **else** **return** 0;  } | Ответ: PPID=1572 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Вылиток А.А. |
| 21 | Что будет выведено на экран, если PID изначально запущенного процесса равен **2143**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else** {wait(NULL);exit(1);}  **}** | Ответ: PPID=2143  Базовое: создание/завершение процесса | Глазкова В.В. |
| 22 | Что будет выведено на экран, если PID изначально запущенного процесса равен **9111**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) printf ( "PID = %d \n", getpid() );**  **else** exit(1);  **}** | Ответ: PID=9111  Базовое: создание/завершение процесса | Глазкова В.В. |
| 23 | Что будет выведено на экран, если PID изначально запущенного процесса равен **315**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) exit(0);**  **else** printf ( "PPID = %d \n", getppid() );  **}** | Ответ: PPID=315 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Глазкова В.В. |
| 24 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **int main()**  **{**  **char buf[2] = “12”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);**  **}** | 7. 12 либо 21 | Дряженков А.А. |
| 25 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **int main()**  **{**  **char buf[3] = “321”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 3);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);**  **}** | 15. 32 либо 23 | Дряженков А.А. |
| 26 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **int main()**  **{**  **char buf[4] = “3210”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 4);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 2);**  **printf(“%c”, buf[0]);**  **return 0;**  **}** | 23. 31 либо 13 | Дряженков А.А. |
| 27 | Что будет выведено на экран, если PID изначально запущенного процесса равен 555? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else** exit();  **}** | Ответ: PPID=555 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Полякова И.Н. |
| 28 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else** {**printf ( "PID = %d \n", getpid() ); exit();}**  **}** | Ответ: PPID=1277  PID=1277  или  PID=1277  PPID=1277  или  PID=1277  PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Полякова И.Н. |
| 29 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( ( pid=fork()) == 0 ) printf ( "PPID = %d \n PID = %d \n ", getppid(), pid )**  **else** **exit();**  **}** | Ответ: PPID=1277  PID=0  Или PPID=1  PID=0   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Полякова И.Н. |
| 30 | Что будет выведено на экран, если PID изначально запущенного процесса равен **3541**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )**  **else** exit();  **}** | Ответ: PPID=3541 или PPID=1 | Жуков К.А. |
| 31 | Что будет выведено на экран, если PID изначально запущенного процесса равен **97123**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )**  **else** exit();  **}** | Ответ: PPID=97123 или PPID=1 | Жуков К.А. |
| 32 | Что будет выведено на экран, если PID изначально запущенного процесса равен **7531**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )**  **else** exit();  **}** | Ответ: PPID=7531 или PPID=1 | Жуков К.А. |
| 33 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1300**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )**  **else** exit(); wait(NULL);  **}** | Ответ: PPID=1300 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Шестимеров А.А. |
| 34 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1275**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) != 0 )** exit();  **printf ( "PPID = %d \n", getppid() )**  **}** | Ответ: PPID=1275 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Шестимеров А.А. |
| 35 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) != 0 ) printf ( "PID = %d \n", getpid() )**  **else** exit();  **}** | Ответ: PPID=1277  Базовое: создание/завершение процесса | Шестимеров А.А. |
| 36 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() == 0 ) printf ( "PPID = %d \n", getppid() );**  **else** {wait(NULL); exit(-1);}  **}** | Ответ: PPID=1277  Базовое: создание/завершение процесса | Сучков Е.П. |
| 37 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() > 0 ) printf ( "PPID = %d \n", getpid() );**  **else** exit(-1);  **}** | 1277 | Сучков Е.П. |
| 38 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1277**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork() > 0 ) exit(-1);**  **printf ( "PPID = %d \n", getppid() );**  **}** | 1 | Сучков Е.П. |
| 39 | Что будет выведено на экран, если PID изначально запущенного процесса равен 111, а PID запущенных процессов - 222 и 333? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main()**  **{**  **if(fork())==0) // 222**  **if (fork()==0){ // 333**  **printf ("PPID1=%d\n",getppid());**  **}else{**  **printf ("PPID2=%d\n",getppid());**  **wait(NULL); exit(0);**  **}**  **else // 111**  **exit(0);**  **}** | PPID1=222  PPID2=111  или  PPID2=111  PPID1=222  или  PPID1=222  PPID2=1  или  PPID2=1  PPID1=222  в зависимости от того, как сработает планировщик  Базовое: создание/завершение процесса | Головин И.Г. |
| 40 | Что будет выведено на экран, если PID изначально запущенного процесса равен 111, а PID запущенных процессов - 222 и 333? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main()**  **{**  **if(fork())==0) // 222**  **if (fork()==0){ // 333**  **printf ("PPID1=%d\n",getppid());**  **}else{**  **printf ("PPID2=%d\n",getppid());**  **wait(NULL); exit(0);**  **}**  **else {// 111**  **wait(NULL);**  **exit(0);**  **}**  **}** | PPID1=222  PPID2=111  или  PPID2=111  PPID1=222  в зависимости от того, как сработает планировщик  Базовое: создание/завершение процесса | Головин И.Г. |
| 41 | Что будет выведено на экран, если PID изначально запущенного процесса равен 111, а PID запущенных процессов - 222 и 333? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main()**  **{ //111**  **if(fork())==0) // 222**  **if (fork()==0){ // 333**  **printf ("PID1=%d\n", getppid());**  **}else{**  **printf ("PID2=%d\n", getppid());**  **}**  **printf ("PID3=%d\n", wait(NULL));**  **exit(0);**  **}** | PID1=222  PID2=111  PID3=-1  PID3=333  PID3=222  в зависимости от того, как сработает планировщик  Базовое: создание/завершение процесса | Головин И.Г. |
| 42 | Что будет выведено на экран, если PID изначально запущенного процесса равен **5229**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено. Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  **main() {**  **int pid;**  **if ( fork()) != 0 ) {printf ( "%d \n", getpid()); exit();}**  **else** { **printf ( "%d \n", getppid() )**};  **}** | Ответ:  5229  5229  Или  5229  1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Семенов А.Н. |
| 43 | Что будет выведено на экран, если PID изначально запущенного процесса равен **1388**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **pid\_t pid;**  **if (pid = fork())) printf ( "PID=%d, PPID = %d \n", pid, getppid() )**  **else** exit();  **}** | Ответ: PID=0,PPID=1388 или PID=0,PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Семенов А.Н. |
| 44 | Что будет выведено на экран, если PID изначально запущенного процесса равен **3386**? Если возможны несколько вариантов – обосновать и привести все варианты. Предполагается, что все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  **main() {**  **int pid;**  **if ( fork()) == 0 ) printf ( "PPID = %d \n", getppid() )**  **else** exit();  **}** | Ответ: PPID=3386 или PPID=1   * в зависимости от того, как сработает планировщик   Базовое: создание/завершение процесса | Семенов А.Н. |

**Задача 8, Базовое: разделение файла при fork**

| **№** | **Условие** | **Ответ** | **Автор** |
| --- | --- | --- | --- |
| 1 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  char buf[3] = “ab”;  int fd = creat(“/a.txt”, 0777);  write(fd, buf, 2);  close(fd);  fd = open(“/a.txt”, O\_RDONLY);  fork();  read(fd, buf, 1);  printf(“%c”, buf[0]);  exit(0); | Ответ: ab либо ba | Базовый |
| 2 | Дан фрагмент программы. Что будет выведено на экран при выполнении этого фрагмента? Если допустимы несколько вариантов вывода приведите их все. Варианты разделить словом «либо».  char buf[3] = "12";  int fd = creat("/a.txt", 0777);  write(fd, buf, 2);  close(fd);  fd = open("/a.txt", O\_RDONLY);  fork();  read(fd, buf, 1);  printf("%c", buf[0]);  exit(0); | 12 или 21 | Казачук М.А. |
| 3 | Что будет выведено на экран при выполнении фрагмента программы? Если допустимы несколько вариантов вывода, приведите все. Считаем, что все системные вызовы отрабатывают полностью и корректно – без отказов.  char buf[5] = "abcf";  int fd = creat("./prob.txt", 0777);  write(fd, buf, 4);  close(fd);  fd = open("./prob.txt", O\_RDONLY);  fork();  read(fd, buf, 2);  printf("%c", buf[1]);  exit(0); | **bf**  либо  **fb** | Казачук М.А. |
| 4 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “ab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ab либо ba  Базовое: разделение файла при форке | Кузина Л.Н. |
| 5 | Какие варианты вывода невозможны при выполнении программы? Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  **int main(){**  **char buf[] = "1234";**  **int fd = open("a.txt", O\_RDWR|O\_CREAT, 0777);**  **write(fd, buf, sizeof(buf));**  **lseek(fd, 2, SEEK\_SET);**  **fork();**  **int fd1 = open("a.txt", O\_RDONLY);**  **read(fd1, buf, 1);**  **printf("%c", buf[0]);**  **read(fd, buf, 1);**  **printf("%c", buf[0]);**  **exit(0);**  **}**  **Варианты вывода:**  1) 1234  2) 1134  3) 1314  4) 1324  5) 4113 | Базовое: разделение файла при форке  невозможны  1) 1234  4) 1324  5) 4113 | Кузина Л.Н. |
| 6 | **Что будет выведено на экран при выполнении фрагмента программы? Если допустимы несколько вариантов вывода, приведите все.**  char buf[5] = “abcd”;  int fd = create (“/a.txt”, 0777);  write (fd, buf, 4);  close (fd);  fd = open (“/a.txt”, O\_RDONLY);  fork ();  fork ();  read (fd, buf, 1);  if (buf[0] != 'a' ) printf (“%c”, buf[0]);  exit (0); | bcd или  cbd или  dbc или  bdc или  cdb или  dcb или | Волкова И.А. |
| 7 | **Что будет выведено на экран при выполнении фрагмента программы? Если допустимы несколько вариантов вывода, приведите все.**  **char** buf[5] = “abcd”;  **int** fd = create (“/a.txt”, 0777);  write (fd, buf, 4);  close (fd);  fd = open (“/a.txt”, O\_RDONLY);  if (fork () == 0) {  read (fd, buf, 1);  printf (“%c”, buf[0]);  }  read (fd, buf, 1);  printf (“%c”, buf[0]);  exit (0); | abc или  bac или  acb | Волкова И.А. |
| 8 | **Что будет выведено на экран при выполнении фрагмента программы? Если допустимы несколько вариантов вывода, приведите все.**  **char** buf[5] = “abcd”;  **int** fd = create (“/a.txt”, 0777);  write (fd, buf, 4);  close (fd);  fd = open (“/a.txt”, O\_RDONLY);  if (fork () == 0) {  read (fd, buf, 1);  printf (“%c”, buf[0]);  exit (0);  }  read (fd, buf, 1);  printf (“%c”, buf[0]);  exit (0); | ab или ba | Волкова И.А. |
| 9 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все. Все системные вызовы прорабатывают успешно.  **char buf[ ] = “qwerty”;**  **int fd = creat(“a”, 0600);**  **write(fd, buf, 4);**  **close(fd);**  **fd = open(“a”, O\_RDONLY);**  **fork();**  **int i = lseek(fd, 1, SEEK\_CUR);**  **printf(“%c”, buf[i]);**  **exit(0);** | we или ew | Корныхин Е.В. |
| 10 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все. Все системные вызовы прорабатывают успешно.  **char buf[ ] = “knm”;**  **int fd = open(“f”, O\_CREAT | O\_RDWR, 0640);**  **write(fd, buf, 4);**  **lseek(fd, 1, SEEK\_SET);**  **fd = open(“f”, O\_RDONLY);**  **fork();**  **read(fd, buf, 2);**  **printf(“%c”, buf[0]);**  **exit(0);** | km или mk | Корныхин Е.В. |
| 11 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все. Все системные вызовы прорабатывают успешно.  **char buf[ ] = “abcde”;**  **int fd = open(“text”, O\_CREAT | O\_RDWR, 0777);**  **write(fd, buf, 5);**  **lseek(fd, -3, SEEK\_CUR);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | cd либо dc | Корныхин Е.В. |
| 12 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  char buf[4] = “abf”;  int fd = creat(“1.txt”, 0777);  write(fd, buf, 3);  close(fd);  fd = open(“1.txt”, O\_RDONLY);  if( !fork()) { fd2= open(“1.txt”, O\_RDONLY);  read(fd, buf, 1);  printf(“%c”, buf[0]);  read(fd2, buf, 1);  printf(“%c”, buf[0]);  exit(0);  }  wait(NULL);  read(fd, buf, 1);  printf(“%c”, buf[0]);  exit(0); | Ответ: **aab**  Базовое: разделение файла при форке | Тюляева В.В. |
| 13 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “ba”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ab либо ba  Базовое: разделение файла при форке | Гомзин А.Г. |
| 14 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “01”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: 01 либо 10  Базовое: разделение файла при форке | Гомзин А.Г. |
| 15 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[4] = “123”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 3);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: 12 либо 21  Базовое: разделение файла при форке |  |
| 16 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “pf”;**  **int fd;**  **fd= creat(“/txt.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/txt.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: pf либо fp | Никольский И.М. |
| 17 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “kj”;**  **int fd = creat(“/X.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/X.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: kj либо jk | Никольский И.М. |
| 18 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “fr”;**  **int fd = creat(“/qwerty.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/qwerty.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: fr либо rf | Никольский И.М. |
| 19 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  main() {  **char** buf[3] = “a1”;  **int** fd = creat(“/a.txt”, 0777);  write(fd, buf, 2);  close(fd);  fd = open(“/a.txt”, O\_RDONLY);  fork();  read(fd, buf, 1);  printf(“%c”, buf[0]);  exit(0);  } | Ответ: a1 либо 1a  Базовое: разделение файла при форке | Вылиток А.А. |
| 20 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  main() {  **char** buf[3] = “b2”;  **int** fd = creat(“/a.txt”, 0777);  write(fd, buf, 2);  close(fd);  fd = open(“/a.txt”, O\_RDONLY);  fork();  read(fd, buf, 1);  printf(“%c”, buf[0]);  exit(0);  } | Ответ: b2 либо 2b  Базовое: разделение файла при форке | Вылиток А.А. |
| 21 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  main() {  **char** buf[3] = “w1”;  **int** fd = creat(“/a.txt”, 0777);  write(fd, buf, 2);  close(fd);  fd = open(“/a.txt”, O\_RDONLY);  fork();  read(fd, buf, 1);  printf(“%c”, buf[0]);  exit(0);  } | Ответ: w1 либо 1w  Базовое: разделение файла при форке | Вылиток А.А. |
| 22 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[4] = “сab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: aс либо сa  Базовое: разделение файла при форке | Глазкова В.В. |
| 23 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[5] = “dcab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **printf(“%c”, buf[0]);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ddc либо dcd  Базовое: разделение файла при форке | Глазкова В.В. |
| 24 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “mn”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **printf(“%c”, buf[0]);**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: mmnm либо mnmm либо mmmn  Базовое: разделение файла при форке | Глазкова В.В. |
| 25 | Содержимое файла “1.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c = '1';  int fd;    fd = open("1.txt", O\_RDONLY);  if(fork())  {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  else  { read(fd, &c, 1); write(1, &c, 1); }  return 0;  } | 8. 114 либо 314 | Дряженков А.А. |
| 26 | Содержимое файла “1.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd = open("1.txt", O\_RDONLY);  if (!fork()) {  read(fd, &c, 1);  write(1, &c, 1);  } else {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 2, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  return 0;  } | 16. 114 либо 314 | Дряженков А.А. |
| 27 | Содержимое файла “1.txt” – строка «12345». Что будет выведено на экран? Если возможны несколько вариантов – привести все.  Предполагается, что обращение к функции вывода на экран прорабатывает атомарно и без буферизации.  Все системные вызовы прорабатывают успешно. Подключение заголовочных файлов опущено.  int main()  {  char c;  int fd = open("1.txt", O\_RDONLY);  if (!fork()) {  read(fd, &c, 1);  write(1, &c, 1);  } else {  int fd2 = open("1.txt", O\_RDONLY);  int fd3 = dup(fd);  lseek(fd, 3, SEEK\_CUR);  wait(NULL);  read(fd2, &c, 1); write(1, &c, 1);  read(fd3, &c, 1); write(1, &c, 1);  }  return 0;  } | 24. 115 либо 415 | Дряженков А.А. |
| 28 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “34”;**  **int d = open(“/t.txt”, O\_WRONLY);**  **write(d, buf, 2);**  **close(d);**  **d = open(“/t.txt”, O\_RDONLY);**  **fork();**  **read(d, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: 34 либо 43  Базовое: разделение файла при форке | Полякова И.Н. |
| 29 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[4] = “cab”;**  **int fd = creat(“/a.txt”, 0666);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ac либо ca  Базовое: разделение файла при форке | Полякова И.Н. |
| 30 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[5] = “abcd”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 5);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 2);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ac либо ca  Базовое: разделение файла при форке | Полякова И.Н. |
| 31 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “xy”;**  **int fd = creat(“/1.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/1.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **putchar(buf[0]);**  **exit(0);** | Ответ: xy либо yx | Жуков К.А. |
| 32 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “12”;**  **int fd = creat(“/b.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/b.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, \*buf);**  **exit(0);** | Ответ: 12 либо 21 | Жуков К.А. |
| 33 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “dg”;**  **int fd = creat(“/2.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/2.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **putchar(\*buf);**  **exit(0);** | Ответ: dg либо gd | Жуков К.А. |
| 34 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “ab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **lseek(fd, 0, SEEK\_SET);**  **exit(0);** | Ответ: ab либо ba либо aa  Базовое: разделение файла при форке | Шестимеров А.А. |
| 35 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “ab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **wait(NULL);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ab либо ba  Базовое: разделение файла при форке | Шестимеров А.А. |
| 36 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “ba”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **lseek(fd, 0, SEEK\_SET);**  **exit(0);** | Ответ: ab либо ba либо bb  Базовое: разделение файла при форке | Шестимеров А.А. |
| 37 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[3] = “pq”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: pq либо qp  Базовое: разделение файла при форке | Сучков Е.П. |
| 38 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[6] = “abcde”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 5);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **if (fork() == 0){**  **read(fd, buf, 2);**  **}**  **else{**  **read(fd, buf, 3);**  **}**  **printf(“%c”, buf[1]);**  **exit(0);** | bd или be | Сучков Е.П. |
| 39 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[6] = “abcde”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 5);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **if (fork() == 0){**  **read(fd, buf, 2);**  **}**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | ad или cd | Сучков Е.П. |
| 40 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[2] = “ab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **if (fork()) {**  **read(fd, buf, 1);**  **} else {**  **lseek(fd, 1, SEEK\_SET);**  **}**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: aa либо ba либо ab  Базовое: разделение файла при форке | Головин И.Г. |
| 41 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[2] = “ab”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **if (fork() == 0) {**  **read(fd, buf, 1);**  **} else {**  **lseek(fd, 1, SEEK\_CUR);**  **}**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ab либо ba либо aa  Базовое: разделение файла при форке | Головин И.Г. |
| 42 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[] = “abcdef”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 2);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **lseek(fd, 0, SEEK\_SET);**  **printf(“%c”, buf[0]);**  **exit(0);** | Ответ: ab либо ba либо aa  Базовое: разделение файла при форке | Головин И.Г. |
| 43 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[5] = “abcd”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 4);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **read(fd, buf, 1);**  **printf(“%c”, buf[0]);**  **exit(0);** | abcd  bacd  abdc  badc  bcad  acdb | Семенов А.Н. |
| 44 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[7] = “abcdef”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 6);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **fork();**  **read(fd, buf, 3);**  **printf(“%c”, buf[1]);**  **exit(0);** | Ответ: be либо eb  Базовое: разделение файла при форке | Семенов А.Н. |
| 45 | Что будет выведено на экран при выполнении программы? Если допустимы несколько вариантов вывода, приведите все.  **char buf[5] = “abcd”;**  **int fd = creat(“/a.txt”, 0777);**  **write(fd, buf, 4);**  **close(fd);**  **fd = open(“/a.txt”, O\_RDONLY);**  **pd = fork();**  **read(fd, buf, 2);**  **printf(“%c%с”, buf[0], buf[1]);**  **exit(0);** | Ответ: abcd либо cdab  Базовое: разделение файла при форке | Семенов А.Н. |
| 46 | #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <fcntl.h>  int main()  {  char buf[] = "1234";  int fd = creat("a.txt", 0777);  write(fd, buf, 4);  close(fd);  fd = open("a.txt", O\_RDONLY);  fork();  read(fd, buf, 2);  printf("%s", buf);  exit(0);  } | /\*  \* '12343434', '34341234'  \*/ | Чернов А.В. |
| 47 | #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <fcntl.h>  int main()  {  char buf[] = "1234";  int fd = creat("a.txt", 0777);  write(fd, buf, 4);  close(fd);  fd = open("a.txt", O\_RDONLY);  read(fd, buf + (fork() > 0), 2);  printf("%s", buf);  exit(0);  } | /\*  \* '11243434', '13441234', '34341124', '12341344'  \*/ | Чернов А.В. |
| 48 | #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <fcntl.h>  int main()  {  char buf[] = "1234";  int fd = creat("a.txt", 0777);  write(fd, buf, 4);  close(fd);  fd = open("a.txt", O\_RDONLY);  read(fd, buf + (fork() > 0), 3);  printf("%s", buf);  exit(0);  } | /\*  \* '11234234', '14341234', '42341123', '12344234'  \*/ | Чернов А.В. |