Операционные системы (филиал МГУ в г. Ереван)

Инструкция к тесту

Данный тест предназначен для проверки знаний по курсу "Операционные системы". Вопросы и задачи разработаны на основе материалов автора курса -- И.В. Машечкина: http://jaffar.cs.msu.ru/mash/os/2021-2022/

Для прохождения теста понадобятся бумага и ручка для промежуточных записей и вычислений.
Время прохождения не ограничено.

Операционные системы (филиал МГУ в г. Ереван)

1 1 из 62

Дан 32-х разрядный IP

адрес, имеющий в восьмеричном представлении вид: $(22011171541)_8$. Определить к какому классу относится данный IP адрес.

B <или> b

2 2 из 62

Дан 32-х разрядный IP-адрес, имеющий в восьмеричном представлении вид: $(22011171541)_8$. Определить номер сети, к которой относится IP-адрес. (Ответ записать в десятичной системе счисления.)

4132

3 3 из 62

Дан 32-х разрядный IP-адрес, имеющий в восьмеричном представлении вид: $(22011171541)_8$. Определить номер хоста, к которому относится данный IP-адрес. (Ответ записать в десятичной системе счисления.)

124613

4 4 из 62

В оперативном запоминающем устройстве 8-ми разрядного встроенного компьютера используется контроль целостности данных по четности. Укажите значение тега контроля четности в случае хранения в машинном слове шестнадцатиричного числа $(A3)_{16}$

0

5 5 из 62

В ОЗУ 16-разрядного компьютера используется контроль целостности данных по четности. Описать побитовое содержимое ячейки памяти с приписанным слева тегом контроля четности для случая хранения в машинном слове восьмеричного числа 17534_8

10001111101011100

6 6 из 62

В оперативном запоминающем устройстве 8-ми разрядного встроенного компьютера используется контроль целостности данных по четности. Укажите значение тега контроля четности в случае хранения в машинном слове шестнадцатиричного числа $(A2)_{16}$

0

7 7 из 62

Пусть дано четверичное число $(133122)_4$, являющееся адресом оперативной памяти, расслоенной по 8 банкам. Укажите номер банка, которому принадлежит заданный адрес. (Ответ записать в десятичной системе счисления)

2

8 8 из 62

Пусть некоторый процесс с pid=2028 породил сыновний процесс с pid=2029, реализованный программой на Си:

```
int  main (int argc, char **argv)
{ if (fork()==0)
    printf (”%d%d”, getpid(), getppid());
 return 0;
}
```

Внутри процесса 2029 порождается процесс с pid=2030. Считаем, что обращения ко всем системным вызовам успешно завершаются. Привести две возможные комбинации значений, которые попадут на стандартное устройство вывода в результате выполнения процессов 2029 и 2030.

20302029

20301

9 9 из 62

Пусть процесс с PID 6121 породил два сыновних проесса с pid-ами 6124 и 6125:

```
int main(int argc, char **argv) // pid = 6121
{
   if (fork() == 0) { // pid = 6124
   printf ("%d",  getpid());
   exit(0);
   }
  wait(NULL);
  if (fork() == 0)  { // pid = 6125
     printf ("%d%d", getpid(), getppid());
     exit(0);
   }
return 0;
}
```

Считаем, что printf работает атомарно и обращения ко всем системным вызовам успешно завершаются. Привести две возможные комбинации значений, которые могут быть выведены на стандартное устройство вывода в результате выполнения данной программы.

612461256121

612461251

10 10 из 62

Каждое ли физическое устройство представлено в системе и как байт-ориентированное устройство и как блок-ориентированное?

- [ ] да
- [x] нет

11 11 из 62

Верно ли, что если физическое устройство блок-ориентированное, то оно и байт-ориентированное ?

- [ ] да
- [x] нет

12 12 из 62

Дана файловая система, имеющая организацию в виде связанного списка. Пусть ссылка в файловом блоке занимает 8 байтов, а размер файлового блока равен 2048 байтов. Пусть некоторый файл из данной файловой системы содержит текстовую информацию (последовательность байтов, содержащих коды символов). Каково минимальное количество обменов можно прочесть часть текста с 2037 символа до 3052 (считаем, что нумерация символов в тексте начинается с 1)?

2

13 13 из 62

Что будет выведено в стандартный поток вывода? 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[] = "df";
        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;
}
```

- [x] dfdf
- [x] fddf
- [ ] fffd
- [ ] fdfd
- [ ] ddf
- [ ] fd

14 14 из 62

В 32-разрядном компьютере реализована двухуровневая таблица страниц. Размер страницы 2048 байтов. Размер каждой таблицы второго уровня равен 8192 записи. Сколько записей содержит «внешняя таблица страниц»?

256

15 15 из 62

Пусть легковесные процессы действуют по программе со следующим фрагментом:

```
int sem =1; // пусть это будет семафор
…
while  (sem ==0); // ждем пока семафор поднимут
sem=0; // опускаем семафор
//работаем с разделяемым ресурсом (файлом)
 write(fd, buf, SIZE);
sem=1; // завершили работу с ресурсом, подняли семафор
```

Обеспечивается ли данной программой взаимное исключение процессов при обращении к файлу через дескриптор fd?

- [ ] да
- [x] нет

16 16 из 62

Пусть легковесные процессы действуют по программе со следующим фрагментом:

```
int sem =1; // пусть это будет семафор
…
while  (sem ==0); // ждем пока семафор поднимут
sem=0; // опускаем семафор
//работаем с разделяемым ресурсом (файлом)
 write(fd, buf, SIZE);
sem=1; // завершили работу с ресурсом, подняли семафор
```

Есть ли в данной программе критическая секция?

- [ ] да
- [x] нет

17 17 из 62

Сколько раз система обратится к содержимому индексных дескрипторов при вызове open("/tmp/folder1/file", O_RDWR) ?
Считаем, что ни один из элементов пути к файлу не является символической ссылкой.

4

18 18 из 62

Каково минимальное число обращений системы к содержимому индексных дескрипторов при вызове open("/etc/file", O_RDWR) ?

3

19 19 из 62

Каково максимальное количество обращений системы к содержимому индексных дескрипторов при вызове open("/name1/name2/name3", O_RDONLY) , если известно что во всей файловой системе есть максимум одна символическая ссылка, она не содержит символ '/' и ее имя отлично от ее содержимого.

5

20 20 из 62

Можно ли между двумя процессами, знающих pid'ы друг друга, передавать текстовые данные, используя только механизм сигналов ?

да	нет
☑	☐

21 21 из 62

Можно ли между двумя процессами, знающих ключ к доступному для них средству IPC, получаемому с помощью вызова semget(), передавать текстовые данные, используя только механизм семафоров ?

да	нет
☑	☐

22 22 из 62

С помощью каких средств можно передавать данные между процессами?

- ☑ файлы
- ☑ очереди сообщений
- ☑ семафоры
- ☑ разделяемая память
- ☑ сокеты
- ☑ сигналы
- ☑ неименованные каналы
- ☑ именованные каналы
- ☐ динамическая память (malloc(), calloc(), free())

23 23 из 62

С помощью каких средств можно синхронизировать процессы?

- [x] файлы (flock())
- [x] очереди сообщений
- [x] семафоры
- [ ] разделяемая память
- [x] сокеты
- [x] сигналы
- [x] неименованные каналы
- [x] именованные каналы
- [ ] динамическая память (malloc(), calloc(), free())
- [ ] люминофоры Дейкстры

24 24 из 62

Каково максимальное число процессов, возникающих при выполнении команды

$ ((cat < file1) & sort file2) | wc >> file3

5

25 25 из 62

Отметьте все варианты, которые могут быть выводом в стандартный поток для следующей программы. Все системные вызовы завершаются успешно. Подключение заголовочных файлов опущено.

```
int main()
{
    pid_t pid;
    int fd[2];
    int x ; /*отсутствие инициализации не считается в языке Си ошибкой*/
    pipe(fd);
     write(1,”a”,1);
    if( (pid = fork()) > 0 )
       {  write(1,”b”,1);
          read(fd[0], &x, sizeof(int));
          write(1,”c”,1);
          wait(NULL);
          write(1,”d”,1);
    }
    else { write(1,”e”,1);
           write(fd[1], &x, sizeof(int));
           write(1,”f”,1);
    }

    return 0;
}
```

- [x] *abecfd*
- [x] *aebcfd*
- [x] *abefcd*
- [x] *aebfcd*
- [x] *aefbcd*
- [ ] *aefbdc*
- [ ] *aefcbd*

Отметьте все варианты, которые могут быть выведены в стандартный поток для следующей программы. Все системные вызовы завершаются успешно. Подключение заголовочных файлов опущено.

```
int main()
{
    pid_t pid;
    int fd[2];
    int x ;
    pipe(fd);
     write(1,"a",1);
    if( (pid = fork()) > 0 )
       {  write(1,"b",1);
          read(fd[0], &x, sizeof(int));
          write(1,"c",1);
          kill(pid, SIGKILL);
          wait(NULL); /* точка 3 */
          write(1,"d",1);
    }
    else { write(1,"e",1);
           write(fd[1], &x, sizeof(int));
           write(1,"f",1);
    }

    return 0;
}
```

- [x] *abecfd*
- [x] *aebcfd*
- [x] *abefcd*
- [x] *aebfcd*
- [x] *aefbcd*
- [x] *abecd*
- [x] *aebcd*
- [ ] *aebdc*
- [ ] *aecbd*

27 27 из 62

Какие функции являются системными вызовами или обращаются к ним при выполнении?

- [x] fprintf()
- [x] fork()
- [x] scanf()
- [x] read()
- [x] write()
- [x] malloc()
- [x] open()
- [ ] sprinf()
- [ ] sscanf()
- [ ] tolower()

28 28 из 62

Отметьте верные для ОС Unix утверждения:

- [x] Сокеты могут использоваться и как локальное, и как сетевое средство межпроцессного взаимодействия
- [ ] Сокет представляет собой файл специального типа
- [x] Запись в сокет и чтение из него могут производиться системными вызовами read/write
- [ ] В случае, если сокет закрыт с противоположной стороны, попытка чтения из него приведет к ошибке
- [ ] При использовании потоковых сокетов любой из двух участников может инициировать подключение
- [x] Дейтаграммные сокеты работают быстрее потоковых, но менее надежны

29 29 из 62

Упорядочите уровни сетевой модели ISO OSI (Open Systems Interconnection) от нижнего к верхнему.

Названия уровней		Номера уровней	
представительский	6	1	первый
физический	1	2	второй
сеансовый	5	3	третий
сетевой	3	4	четвертый
канальный	2	5	пятый
прикладной	7	6	шестой
транспортный	4	7	седьмой

Установите соответствие между названиями и определениями терминов ОС

Названия терминов	
Файл	1
Индексный дескриптор	2
Каталог	3
Файловый дескриптор	4
Открытие файла	5

	Определения терминов
1	специальным образом проименованный набор данных, размещенный в системе
2	системная структура данных, содержащая информацию о размещении блоков конкретного файла в файловой системе
3	компонент файловой системы с информацией о содержащихся в файловой системе файлов (является специальным видом файла)
4	системная структура данных, содержащая информацию об актуальном состоянии открытого файла
5	регистрация в системе возможности работы процесса с содержимым файла

Установите соответствие между названиями и определениями терминов ОС Unix

Механизм сигналов	1
Именованный канал	2
Сокет	3
IPC	4
Неименованный канал	5
Трассировка процессов	6
Очередь сообщений	7
Разделяемая память	8
Семафор	9
Двоичный семафор Дейкстры	10
Счетчиковый семафор	11
Монитор Хоара	12

1 средство уведомления процесса о наступлении одного из событий в системе (перечень возможных событий зафиксирован в системе).

2 отдельный тип файла в файловой системе Unix (обадающий всеми атрибутами файла, такими как имя владельца, прав доступа, время модификации и др.) по функционироанию идентичный неменованному каналу.

3 объект, который может быть именован в зависимости от реализованного коммуникционного домена, выбранного при его создании.

4 средства (очередь сообщений, разделяемая память, семфоры), позволяющие организовать взаимодействие процессов в рамках одного сеанса ОС.

5 сущность, в которую можно помещать и извлекать данные посредством двух ассоциированных файловых дескрипторов: один для записи, другой для чтения.

6 вид межпроцессного взаимодействия по модели "главный-подчиненный", когда один процесс получает возможность управлять ходом выполнения, данными и кодом другого.

7 хранилище типизированных сообщений, организованное по принципу FIFO.

8 механизм, позволяющий нескольким процессам получить отображение некоторых участков из своей виртуальной памяти на общую область физической памяти.

9 особый вид числовой переменной (с неотрицательными значениями), над которой определены 2 **неделимые** операции: уменьшение значения с возможным блокированием процесса (при попытке достичь отрицательного значения) и увеличение значения с возможным разблокированием одного из ранее заблокированных процессов.
В System V IPC это массив таких переменных. -- это двоичный семафор Дейксты, при большем диапазоне значений от 0 до k>1 -- счетчиковый семфор.

10 семафор с двумя возможными значениями: 0 ("шлагбаум закрыт") и 1 ("шлагбаум открыт").

11 семафор со значениями от 0 до k>1.

12 аппарат, обеспечиваемый совокупностью процедур и структур данных, объединенных в программный модуль специального типа; принадлежит только одному процессу; если он занят, то процесс блокируется и ожидает

32 32 из 62

Какие проблемы должен решать менеджер оперативной памяти?

- [x] нехватка объёма оперативной памяти
- [ ] недостаток места на диске
- [x] фрагментация оперативной памяти
- [ ] аппаратные сбои при работе микросхем памяти
- [ ] сохранение и восстановление регистров

33 33 из 62

Вася создал у себя в домашнем каталоге подкаталог in:

drwxr-x--- 125 vasya vasya 340 Sep 28 in

Он хочет, чтобы любой пользователь мог скопировать в этот каталог файлы, после чего сам Вася мог бы с ними ознакомиться. Требуется обеспечить невозможность удаления помещенного в этот каталог файла любым пользователем за исключением создателя файла.
Выберите команду, которую должен запустить Вася.

- [ ] chmod a+w+r in
- [x] chmod o-w+t+r in
- [ ] chmod u+r+w+x in
- [ ] chmod g+s in
- [ ] chmod g-w+t+r in

34 34 из 62

Что из перечисленного ниже относится к задачам ядра ОС Unix?

- [x] обеспечение взаимодействия процессов
- [ ] вывод на экран графического пользовательского интерфейса
- [x] управление виртуальной памятью и подкачкой
- [ ] управление кэш-памятью
- [x] управление вводом-выводом

Рассмотрим команду:

$ rm mypipe; mkfifo mypipe; sort < mypipe & cat > mypipe

Что будет выведно на экран этой командой, если после ее запуска набрать на клавиаутре строки

aa

ssd

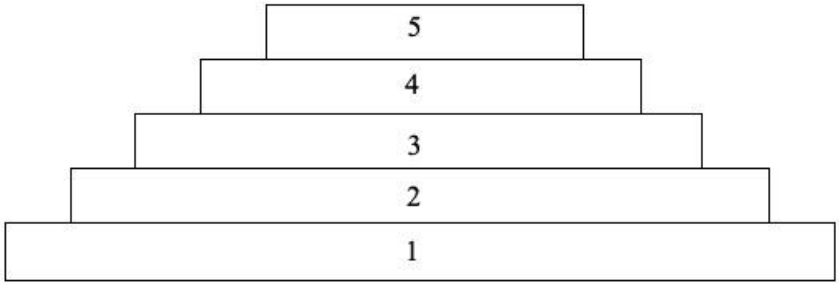
bc

<Ctrl-D>

?

- [x] aa
  bc
  ssd
- [ ] 1 aa
  2 bc
  3 ssd
- [ ] aa
  ssd
  bc
- [ ] ssd
  bc
  ssd
- [ ] ничего не будет выведено

Структура вычислительной системы



Вычислительная система — совокупность аппаратных и программных средств,функционирующих в единой системе и предназначенных для решения задач определенного класса.

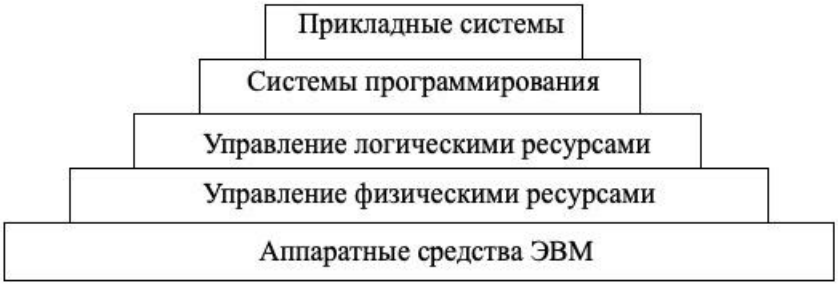
Струкутуру вычислительной сиситмы можно представить в виде пирамиды из уровней от первого (низшего) до пятого (высшего).

Соотнесите уровни с их порядковым номером.

1	аппартные средства ЭВМ
2	управление физическими ресурсами
3	управление логическими ресурсами
4	системы программирования
5	прикладные системы

37 37 из 62

Структура вычислительной системы



Какие уровни вычислительной системы относятся к операционной системе?

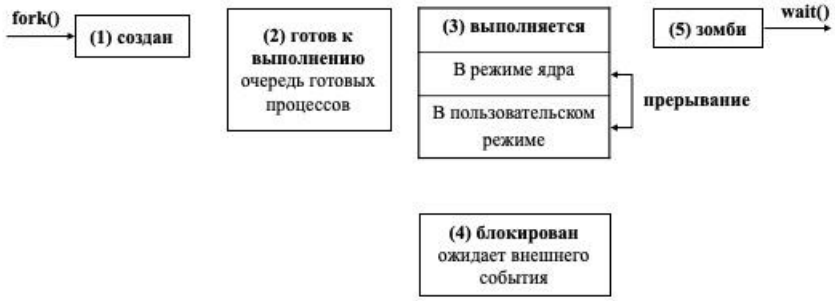
- [ ] Прикладные системы
- [ ] Системы программирования
- [x] Управления логическими ресурсами
- [x] Управление физическими ресурсами
- [ ] Аппаратные срдества ЭВМ

38 38 из 62

Пусть в файловой системе ОС Unix System V размер блока равен 512 байт, а для указания номера блока используется тип int (размером 4 байта).

Какова максиально возможная длина файла в блоках данной системе?

2113674



Дана заготовка схемы жизненного цикла процесса в ОС Unix. Укажите все возможные переходы между состояниями в схеме жизненного цикла.

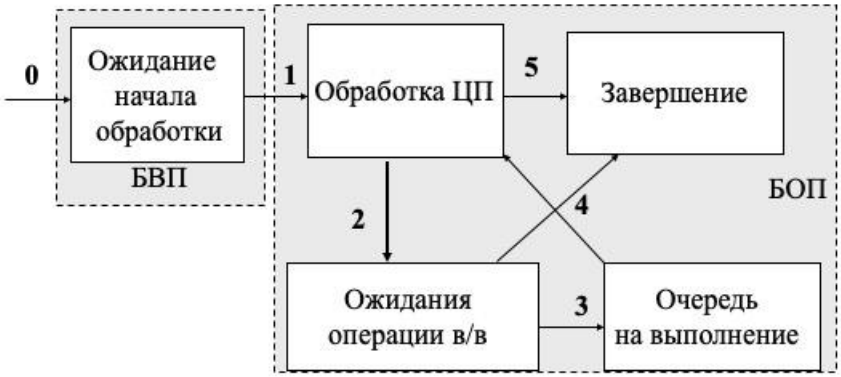
- [x] 1-->2
- [ ] 2-->1
- [x] 2-->3 планировщик
- [x] 3-->2 истек квант времени
- [x] 3-->4
- [ ] 4-->3
- [x] 4-->2 внешнее событие
- [x] 3-->5 exit()
- [ ] 4-->5
- [ ] 1-->5

0. Поступление процесса в очередь на начало обработки ЦП (процесс попадает в БВП)
1. Начало обработки процесса на ЦП (из БВП в БОП)
2. Завершение выполнения процесса, освобождение системных ресурсов.

Жизненный цикл какой модели ОС изображен на данной схеме?

Буфер ввода процессов (БВП) — пространство, в котором размещаются и хранятся сформированные процессы с момента их образования, до момента начала выполнения.
Буфер обрабатываемых процессов (БОП) — буфер для размещения процессов, находящихся в системе в мультипрограммной обработке.

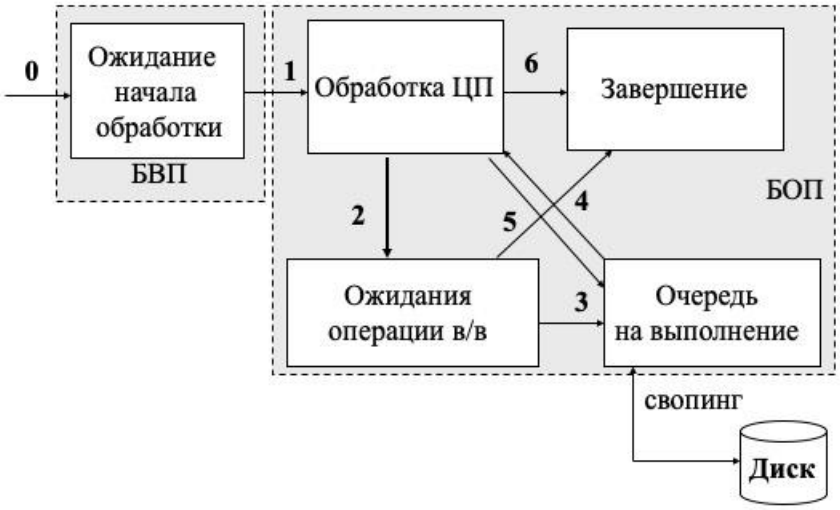
- [x] пакетная однопроцессная система
- [ ] пакетная мультипроцессная система
- [ ] система с разделением времени
- [ ] система с разделением времени и подкачкой



Жизненный цикл какой модели ОС изображен на данной схеме?

Буфер ввода процессов (БВП) — пространство, в котором размещаются и хранятся сформированные процессы с момента их образования, до момента начала выполнения.
Буфер обрабатываемых процессов (БОП) — буфер для размещения процессов, находящихся в системе в мультипрограммной обработке.

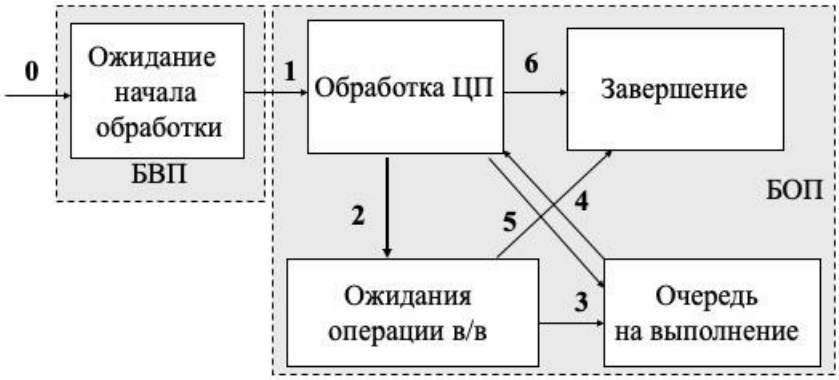
- [ ] пакетная однопроцессная система
- [x] пакетная мультипроцессная система
- [ ] система с разделением времени
- [ ] система с разделением времени и подкачкой



Жизненный цикл какой модели ОС изображен на данной схеме?

Буфер ввода процессов (БВП) — пространство, в котором размещаются и хранятся сформированные процессы с момента их образования, до момента начала выполнения.
Буфер обрабатываемых процессов (БОП) — буфер для размещения процессов, находящихся в системе в мультипрограммной обработке.

- [ ] пакетная однопроцессная система
- [ ] пакетная мультипроцессная система
- [ ] система с разделением времени
- [x] система с разделением времени и подкачкой



Жизненный цикл какой модели ОС изображен на данной схеме?

Буфер ввода процессов (БВП) — пространство, в котором размещаются и хранятся сформированные процессы с момента их образования, до момента начала выполнения.
Буфер обрабатываемых процессов (БОП) — буфер для размещения процессов, находящихся в системе в мультипрограммной обработке.

- [ ] пакетная однопроцессная система
- [ ] пакетная мультипроцессная система
- [x] система с разделением времени
- [ ] система с разделением времени и подкачкой

44 44 из 62

Возможна ли тупиковая ситуация в данном решении задачи о пяти обедающих философах, когда i-й философ представлен легковесным процессом и выпоняет функцию philosopher(i) ?

```
#define N 5
semaphore s=4;
...
void philosopher ( int i )
{
   while (TRUE) {
      think () ;
      down(&s);
      take_fork ( i ) ;
      take_fork ( ( i + 1 ) % N ) ;
      eat () ;
      put_fork ( i ) ;
      put_fork ( ( i + 1 )  % N ) ;
      up(&s);
   }
   return;
}
```

- ○ да
- ✓ нет

45 45 из 62

Возможна ли тупиковая ситуация в данном решении задачи о пяти обедающих философах, когда i-й философ представлен легковесным процессом и выпоняет функцию philosopher(i) ?

```
#define N 5
...
void philosopher ( int i )
{
   while (TRUE) {
      think () ;
      take_fork ( i ) ;
      take_fork ( ( i + 1 ) % N ) ;
      eat () ;
      put_fork ( i ) ;
      put_fork ( ( i + 1 )  % N ) ;
   }
   return;
}
```

- ✓ да
- ○ нет

Заполните пропуски (без лишних пробелов) в решении Дейкстры задачи о пяти обедающих философах.

```
# define  N  5
# define  LEFT  ( i – 1 ) % N
# define  RIGHT  ( i + 1 ) % N
# define  THINKING  0
# define  HUNGRY  1
# define  EATING  2

typedef  int  semaphore ;
int state [ N ] ;
semaphore mutex = 1 ;
semaphore s [ N ] ;
void philosopher ( int i )
{  while ( TRUE )
  {
  think () ;
  take_forks ( i ) ;
  eat ();
  put_forks ( i ) ;
  }
}
void take_forks ( int i )
{
  down ( &mutex ) ;
  state [ i ] =  HUNGRY ;
  test ( i ) ;
  up( &mutex ) ;
  down (    &s[I]    ) ;
}
void put_forks ( int i )
{
  down ( & mutex ) ;
  state[i] = THINKING ;
  test ( LEFT ) ;
  test ( RIGHT ) ;
  up (  &mutex  ) ;
}
void test ( int i )
{
  if ( ( state [ i ] == HUNGRY )   && ( state [ LEFT ] != EATING )
  && ( state [ RIGHT ] != EATING ))
  {
    state [ i ] = EATING ;
  up ( &s [ i ] ) ;
  }
}
```

Заполните пропуски (без лишних пробелов) в решении Дейкстры задачи о пяти обедающих философах.

```
# define  N  5
# define  LEFT  ( i – 1 ) % N
# define  RIGHT  ( i + 1 ) % N
# define  THINKING  0
# define  HUNGRY  1
# define  EATING  2

typedef  int  semaphore ;
int state [ N ] ;
semaphore mutex = 1 ;
semaphore s [ N ] ;
void philosopher ( int i )
{  while ( TRUE )
  {
  think () ;
  take_forks ( i ) ;
  eat ();
  put_forks ( i ) ;
  }
}
void take_forks ( int i )
{
  down (  &mutex  ) ;
  state [ i ] =  HUNGRY ;
  test ( i ) ;
  up( & mutex ) ;
  down ( &s[i] ) ;
}
void put_forks ( int i )
{
  down ( &mutex ) ;
  state[i] = THINKING ;
  test ( LEFT ) ;
  test ( RIGHT ) ;
  up ( &mutex ) ;
}
void test ( int i )
{
  if ( ( state [ i ] == HUNGRY )   && ( state [ LEFT ] != EATING )
  && ( state [ RIGHT ] != EATING ))
  {
    state [ i ] = EATING ;
    up (    &s[I]    ) ;
  }
}
```

Заполните пропуски (без лишних пробелов) в решении задачи о читателях и писателях.

```
typedef int semaphore ;
int rc = 0 ;
semaphore mutex = 1 ;
semaphore db = 1 ;

void reader ( void )
{
    while ( TRUE ){
        down ( &mutex ) ;
        rc++ ;
        if ( rc == 1 ) down ( &db ) ;
        up ( & mutex ) ;
        read_data_base () ;
        down ( & mutex ) ;
        rc— ;
        if ( rc ==0 ) up ( & db ) ;
        up ( &mutex ) ;
        use_data_read () ;
     }
}

void writer ( void )
{
  while ( TRUE )
  {
  think_up_data () ;
     down ( &db ) ;
  write_data_base () ;
     up ( & db ) ;
  }
}
```

49 49 из 62

Заполните пропуски (без лишних пробелов) в решении задачи о спящем парикмахере.

```
#define CHAIRS 5
typedef int semaphore ;
semaphore customers = 0 ;
semaphore barber = 0 ;
semaphore mutex = 1 ;
int waiting = 0 ;

void barber ( void )
{
    while ( TRUE ) {
        down ( &customers ) ;
        down ( &mutex ) ;
        waiting = wating – 1 ;
        up ( &barber ) ;
        up ( &mutex ) ;
        cut_hair () ;
    }
}

void customer ( void )
{
    down ( &mutex ) ;
    if ( waiting < CHAIRS ) {
        waiting = waiting + 1 ;
        up ( &customers ) ;
        up ( &mutex ) ;
        down ( barber ) ;
        get_haircut () ;
    }
    else {
        up ( &mutex ) ;
    }
}
```

50 50 из 62

Пусть головка HDD позиционирована на дорожке 15 и существует очередь запросов к дорожкам с номерами: 4, 40, 11, 35, 7, 14.

Какова общая длина пермещений головки в случае использования алгоритма FIFO для планирования дисковых обменов?

135

51 51 из 62

Пусть головка HDD позиционирована на дорожке 15 и существует

очередь запросов к дорожкам с номерами: 4, 40, 11, 35, 7, 14.

Какова общая длина пермещений головки в случае использования алгоритма LIFO для планирования дисковых обменов?

125

52 52 из 62

Пусть головка HDD позиционирована на дорожке 15 и существует

очередь запросов к дорожкам с номерами: 4, 40, 11, 35, 7, 14.

Какова общая длина пермещений головки в случае использования «жадного» алгоритма Shortest Service Time First (SSTF) для планирования дисковых обменов?

47

53 53 из 62

Пусть головка HDD позиционирована на дорожке 15 и существует

очередь запросов к дорожкам с номерами: 4, 40, 11, 35, 7, 14.

Какова общая длина пермещений головки в случае использования «лифтового» алгоритма SCAN для планирования дисковых обменов? Максимальный номер доржки – 40.

61

54 54 из 62

Пусть головка HDD позиционирована на дорожке 15 и существует

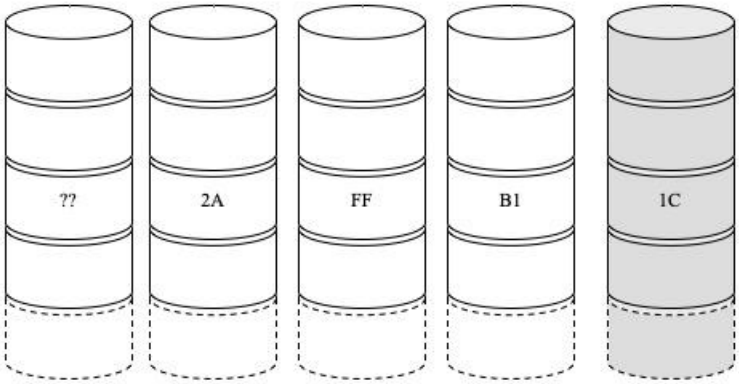
очередь запросов к дорожкам с номерами: 4, 40, 11, 35, 7, 14.

Какова общая длина пермещений головки в случае использования алгоритма циклического сканирования C-SCAN для планирования дисковых обменов?

Максимальный номер дорожки – 40.

47

55 55 из 62



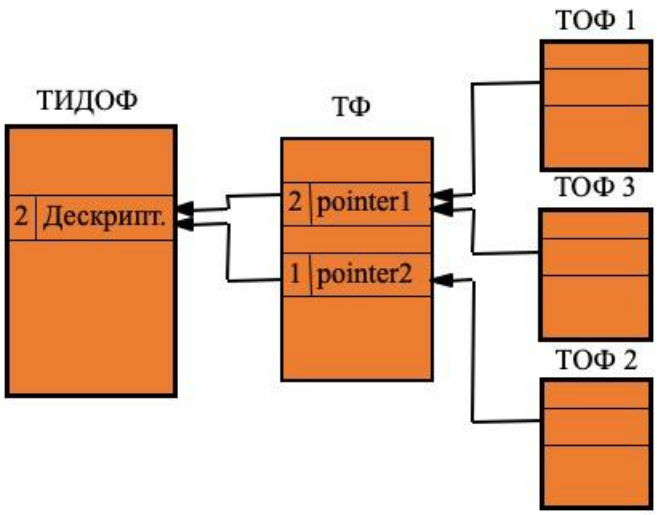
RAID (Redundant Array of Independent (Inexpensive) Disks) — избыточный массив независимых (недорогих) дисков.

RAID система — набор физических дисковых устройств, рассматриваемых операционной системой, как единое дисковое устройство (данные распределяются по физическим устройствам, образуется избыточная информация, используемая для контроля и восстановления информации).

Пусть используется система RAID_3. Восстановите потерянный байт на нулевом диске.

78

56 56 из 62



Укажите номера схем программ, приводящих к следующей конфигурации содержимого системных таблиц (полагаем, что других системных вызоовов fork(), open() и dup() в программе нет):

(1)

```
...
 if ( fork() == 0)  { fd=open(filename, O_RDONLY);  .... ; exit(0);}
 fd = open(filename, O_RDONLY);
 if (fork() == 0) { .... ; exit(0); }
...
```

(2)

```
...
 fd = open(filename, O_RDONLY);
 if (fork() == 0) {  .... ; exit(0); }
 if ( fork() == 0)  { fd=open(filename, O_RDONLY); ... ; exit(0); }
...
```

(3)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=dup(fd1);
...
```

(4)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=open(filename, O_RDONLY);
....
```

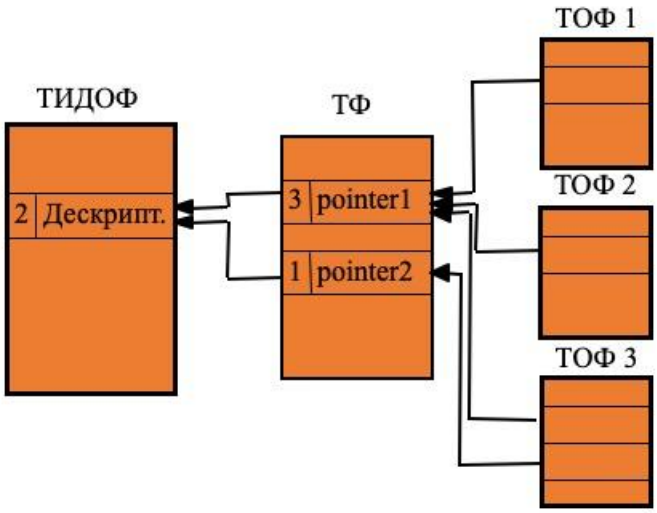
(5)

```
...
fd1=open(filename, O_RDONLY);
fd2=dup(fd1);
if (fork()==0)  { .... ; exit(0); }
....
```

(6)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { fd2=dup(fd1);  .... ; exit(0); }
fd2=dup(fd1);
...
```

- [x] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5
- [ ] 6


ТОФ 1
ТИДОФ
ТФ
ТОФ 2
2 Дескрипт.
3 pointer1
1 pointer2
ТОФ 3

Укажите номера схем программ, приводящих к следующей конфигурации содержимого системных таблиц (полагаем, что других системных вызоовов fork(), open() и dup() в программе нет):

(1)

```
...
 if ( fork() == 0)  { fd=open(filename, O_RDONLY);  .... ; exit(0);}
 fd = open(filename, O_RDONLY);
 if (fork() == 0) { .... ; exit(0); }
...
```

(2)

```
...
 fd = open(filename, O_RDONLY);
 if (fork() == 0) {  .... ; exit(0); }
 if ( fork() == 0)  { fd=open(filename, O_RDONLY); ... ; exit(0); }
...
```

(3)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=dup(fd1);
...
```

(4)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=open(filename, O_RDONLY);
....
```

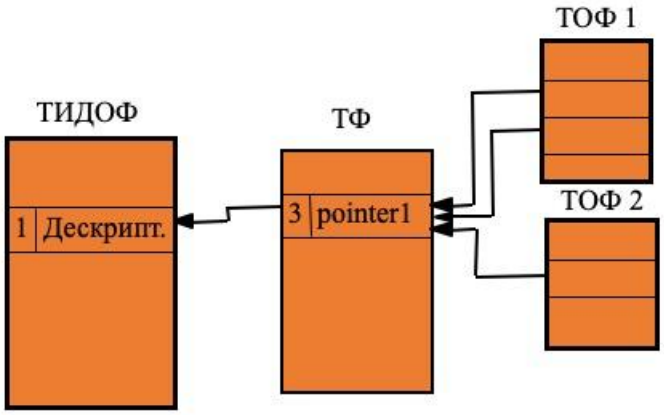
(5)

```
...
fd1=open(filename, O_RDONLY);
fd2=dup(fd1);
if (fork()==0)  { .... ; exit(0); }
....
```

(6)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { fd2=dup(fd1);  .... ; exit(0); }
fd2=dup(fd1);
...
```

- [ ] 1
- [x] 2
- [ ] 3
- [ ] 4
- [ ] 5
- [ ] 6


ТОФ 1
ТИДОФ
ТФ
1 Дескрипт.
3 pointer1
ТОФ 2

Укажите номера схем программ, приводящих к следующей конфигурации содержимого системных таблиц (полагаем, что других системных вызоовов fork(), open() и dup() в программе нет):

(1)

```
...
 if ( fork() == 0)  { fd=open(filename, O_RDONLY);  .... ; exit(0);}
 fd = open(filename, O_RDONLY);
 if (fork() == 0) { .... ; exit(0); }
...
```

(2)

```
...
 fd = open(filename, O_RDONLY);
 if (fork() == 0) {  .... ; exit(0); }
 if ( fork() == 0)  { fd=open(filename, O_RDONLY); ... ; exit(0); }
...
```

(3)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=dup(fd1);
...
```

(4)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=open(filename, O_RDONLY);
....
```

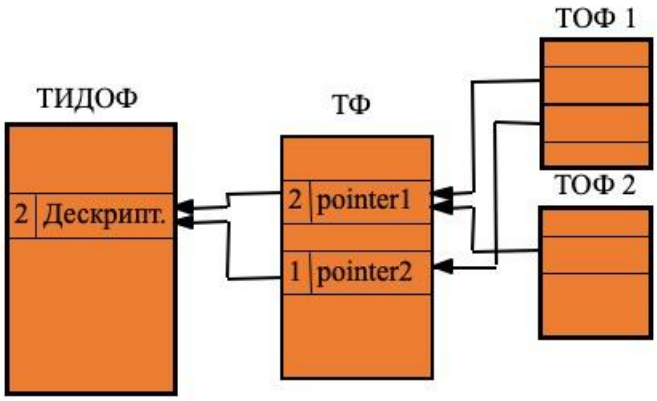
(5)

```
...
fd1=open(filename, O_RDONLY);
fd2=dup(fd1);
if (fork()==0)  { .... ; exit(0); }
....
```

(6)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { fd2=dup(fd1);  .... ; exit(0); }
fd2=dup(fd1);
...
```

- [ ] 1
- [ ] 2
- [x] 3
- [ ] 4
- [ ] 5
- [ ] 6


ТОФ 1
ТИДОФ
ТФ
ТОФ 2
2 Дескрипт.
2 pointer1
1 pointer2

Укажите номера схем программ, приводящих к следующей конфигурации содержимого системных таблиц (полагаем, что других системных вызоовов fork(), open() и dup() в программе нет):

(1)

```
...
 if ( fork() == 0)  { fd=open(filename, O_RDONLY);  .... ; exit(0);}
 fd = open(filename, O_RDONLY);
 if (fork() == 0) { .... ; exit(0); }
...
```

(2)

```
...
 fd = open(filename, O_RDONLY);
 if (fork() == 0) {  .... ; exit(0); }
 if ( fork() == 0)  { fd=open(filename, O_RDONLY); ... ; exit(0); }
...
```

(3)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=dup(fd1);
...
```

(4)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=open(filename, O_RDONLY);
....
```

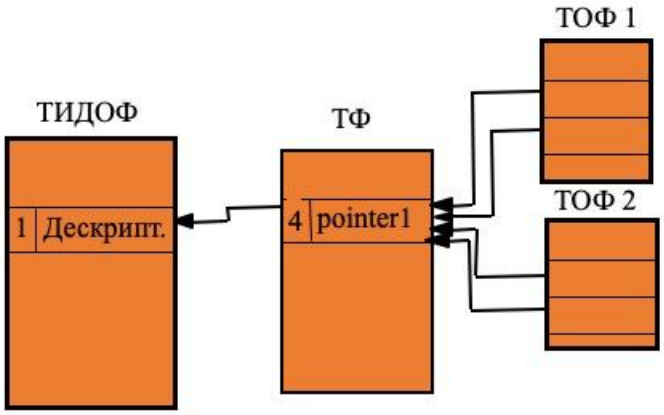
(5)

```
...
fd1=open(filename, O_RDONLY);
fd2=dup(fd1);
if (fork()==0)  { .... ; exit(0); }
....
```

(6)

```
...
fd1=open(filename, O_RDONLY);
if (fork()==0)  { fd2=dup(fd1);  .... ; exit(0); }
fd2=dup(fd1);
...
```

- [ ] 1
- [ ] 2
- [ ] 3
- [x] 4
- [ ] 5
- [ ] 6


ТОФ 1
ТИДОФ
ТФ
1 Дескрипт.
4 pointer1
ТОФ 2

Укажите номера схем программ, приводящих к следующей конфигурации содержания системных таблиц (полагаем, что других системных вызоовов fork(), open() и dup() в программе нет):

(1)

...

```
if ( fork() == 0)  { fd=open(filename, O_RDONLY);  .... ; exit(0);}
fd = open(filename, O_RDONLY);
if (fork() == 0) { .... ; exit(0); }
```

...

(2)

...

```
fd = open(filename, O_RDONLY);
if (fork() == 0) {  .... ; exit(0); }
if ( fork() == 0)  { fd=open(filename, O_RDONLY); ... ; exit(0); }
```

...

(3)

...

```
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=dup(fd1);
```

...

(4)

...

```
fd1=open(filename, O_RDONLY);
if (fork()==0)  { .... ; exit(0); }
fd2=open(filename, O_RDONLY);
....
```

(5)

...

```
fd1=open(filename, O_RDONLY);
fd2=dup(fd1);
if (fork()==0)  { .... ; exit(0); }
....
```

(6)

...

```
fd1=open(filename, O_RDONLY);
if (fork()==0)  { fd2=dup(fd1);  .... ; exit(0); }
fd2=dup(fd1);
```

...

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [x] 5
- [x] 6

Какое максимальное количество сокетов может существовать во время обмена через виртуальный канал с одним клиентским процессом при работе программы по указанной ниже схеме. (Заголовочные файлы опущены; других системных вызовов (кроме указанных в схеме), относящихся к сокетам, нет).

```
int main ( int argc, char ** argv )
{   struct sockaddr_in own_addr, party_addr ;
  int sockfd, newsockfd, filefd ;
    int party_len ;
    char buf [ BUFLEN ] ;
  int len ;
    int i ;
  sockfd = socket ( AF_INET, SOCK_STREAM, 0 );
     …
  memset ( & own_addr, 0, sizeof ( own_addr ) ) ;
  own_addr . sin_family = AF_INET ; own_addr . sin_addr . s_addr = INADDR_ANY ; own_addr . sin_port = htons ( PORTNUM ) ;
     bind ( sockfd, ( struct sockaddr * ) & own_addr, sizeof ( own_addr ) ) ;
  listen ( sockfd, BACKLOG ) < 0 )  ;
  …
  while ( 1 ) {
    memset ( & party_addr, 0, sizeof ( party_addr ) ) ;
    party_len = sizeof ( party_addr ) ; /* создаем соединение */
    newsockfd = accept ( sockfd, ( struct sockaddr * ) & party_addr, & party_len ) ;
    if ( ! fork () ) {/*это — сын*/
          close ( sockfd ) ; /*  этот сокет сыну не нужен */
          … /* обмен сообщениями с клиентом через newsockfd */
          shutdown ( newsockfd, 1 ) ;
          close ( newsockfd ) ;
          return 0 ;
     }
   /*это — отец*/
     …
   shutdown ( newsockfd, 1 ) ;
   close ( newsockfd ) ;
   return 0 ;
   }
   …
}
```

2

62 62 из 62

Какое минимальное количество сокетов может быть у главного сереверного процесса во время обслуживания одного клиентского процесса сервисным процессом-сыном при работе программы по указанной ниже схеме. (Заголовочные файлы опущены; других системных вызовов (кроме указанных в схеме), относящихся к сокетам, нет).

```
int main ( int argc, char ** argv )
{   struct sockaddr_in own_addr, party_addr ;
  int sockfd, newsockfd, filefd ;
    int party_len ;
    char buf [ BUFLEN ] ;
  int len ;
    int i ;
  sockfd = socket ( AF_INET, SOCK_STREAM, 0 );
     …
  memset ( & own_addr, 0, sizeof ( own_addr ) ) ;
  own_addr . sin_family = AF_INET ; own_addr . sin_addr . s_addr = INADDR_ANY ; own_addr . sin_port = htons ( PORTNUM ) ;
     bind ( sockfd, ( struct sockaddr * ) & own_addr, sizeof ( own_addr ) ) ;
  listen ( sockfd, BACKLOG ) < 0 )  ;
  …
  while ( 1 ) {
    memset ( & party_addr, 0, sizeof ( party_addr ) ) ;
    party_len = sizeof ( party_addr ) ; /* создаем соединение */
    newsockfd = accept ( sockfd, ( struct sockaddr * ) & party_addr, & party_len ) ;
    if ( ! fork () ) {/*это — сын*/
          close ( sockfd ) ; /*  этот сокет сыну не нужен */
          … /* обмен сообщениями с клиентом через newsockfd */
          shutdown ( newsockfd, 1 ) ;
          close ( newsockfd ) ;
          return 0 ;
     }
   /*это — отец*/
     …
   shutdown ( newsockfd, 1 ) ;
   close ( newsockfd ) ;
   return 0 ;
   }
   …
}
```

1