Коллоквиум по ОС с ответами, 2022 год
Вариант 1

1. В ОЗУ 16-разрядного компьютера используется контроль целостности данных по четности. Описать возможную структуру ячейки памяти и ее побитовое содержимое для случая хранения в машинном слове восьмеричного числа 35735_8.

Ответ: в 16-разрядном компьютере ячейка памяти содержит 16 бит данных, бит паритета (тег) – дополнительный, он в данные не входит.
Структура ячейки памяти: **0011 1011 1101 1101** – это данные, и еще один бит (приписывается справа или слева) равен **1** – он вычисляется как сумма по модулю 2 (`xor`) всех битов данных.

2. Пусть дано четверичное число 123123_4, являющееся адресом оперативной памяти, расслоенной по 8 банкам. Банку с каким номером принадлежит заданный адрес?

Ответ: **3**. В схеме расслоения памяти последовательные адреса размещаются в последовательных банках памяти. При 8 банках памяти за номер банка будут отвечать младшие 3 бита адреса. Они равны 011 (от четверичной двойки в двоичном виде 10 берется последний ноль, из четверичной тройки получается двоичное 11), то есть 3 в десятичной системе. Банки нумеруются всегда с 0.

3. Пусть некоторый процесс с pid= X породил два сыновних процесса с pid=Y и pid=Z, реализованный программой на Си:

```
int  main (int argc, char **argv) /* pid=X */
{   int pid=getpid();
    if (fork()==0){ /* pid=Y */
        printf (”%d %d\n”, pid,getpid());
        exit(0);
    }
    if (fork()==0){ /* pid=Z */
        printf (”%d\n”,getppid());
        exit(0);
    }
 return 0;
}
```

Считаем, что printf работает атомарно без буферизации и обращения ко всем системным вызовам срабатывают успешно, заголовочные файлы опущены. Перечислить все возможные комбинации значений, которые попадут на стандартное устройство вывода в результате выполнения данной программы.

Ответ: Процесс X не ждет процесс Z (отсутствует wait()), поэтому X может завершиться как до завершения Z (тогда осиротевший Z станет потомком процесса init с pid=1), так и после. Никакой синхронизации между братьями Y и Z нет. Поэтому возможны четыре варианта ответа:

1	2	3	4
X Y **X**	**X Y** **1**	**X** **X Y**	**1** **X Y**

4. Что будет выведено в стандартный поток вывода? Привести все возможные варианты ответа. Считаем, что обращение ко всем системным вызовам срабатывает успешно, а printf работает атомарно и без буферизации. 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;
}
```

Ответ: Точкой синхронизации, где сын ждет отца, является второй read в сыне. До этого момента отец напечатает ”01” , а в канал попадет последовательность ”10” или ”01”. Поэтому возможны два варианта ответа:

1	2
01	**01**
01	**10**

5. Дана файловая система, имеющая организацию в виде связного списка. Пусть ссылка в файловом блоке занимает 4 байта, а размер файлового блока равен 1024 байтам. Пусть некоторый файл из данной файловой системы содержит текстовую информацию (последовательность байтов, содержащих коды символов). За какое минимальное количество обменов можно прочесть часть текста с 2047 символа до 5172 символа (нумерация символов в тексте начинается с 1).

Ответ: Содержательная информация в файловом блоке занимает 1024 – 4 = 1020 байтов. При данной организации файловой системы придется прочитать последовательно все блоки (согласно списку) от первого до блока, содержащего 5172-й символ. Поделив 5172 на 1020, получим 5.071, то есть 5172-й (последний) символ находится в 6-м блоке (округляем 5.071 в большую сторону), то есть требуется выполнить **6 обменов**.

6. Пусть в 32-разрядном компьютере используется страничная память с двухуровневой таблицей страниц. Размер страницы 4096 байт. Таблица страниц первого уровня (внешняя) содержит 8192 записи. Определить размер каждой таблицы второго уровня.

Ответ: Виртуальный адрес – это комбинация номера страницы первого уровня, номера страницы второго уровня и смещения в странице, всего 32 бита.
$4096 = 2^{12}$, следовательно под смещение в странице отводится 12 бит.
$8192 = 2^{13}$, следовательно под номер страницы первого уровня отводится 13 бит.
Под номер страницы второго уровня остается 32-12-13=7 бит.
Следовательно, размер каждой таблицы второго уровня равен 2^7=**128**.

7. Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран срабатывает атомарно и без буферизации. Все системные вызовы срабатывают 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 = 1;
          write(fd[1], &x, sizeof (int));
          x = 2;
     }
     printf("%d", x);
     return 0;
}
```

Ответ: **31** либо **321**. Отцовский процесс блокируется на чтении из канала до тех пор, пока сын не напечатает 3 (начальное значение x) и не запишет в канал 1. Эту единицу отец запишет в свою копию переменной x, пошлёт сыну сигнал и будет ждать его завершения, после чего напечатает 1. Сын может успеть до получения сигнала (приводящего к его завершению) напечатать 2, а может не успеть. Поэтому возможных вариантов ответа два.

8. Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран срабатывает атомарно и без буферизации. Все системные вызовы срабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{    char c='x';
     int fd[2], fd2[2];
     pipe(fd);
     pipe(fd2);
     if(fork() == 0) {
          read(fd[0], &c,1); putchar('d');
          read(fd2[0], &c,1); putchar('b');
          exit(0);
     }
     putchar('a');
     write(fd[1], &c, 1);
     putchar('c');
     write(fd2[1], &c, 1);
     wait(NULL);
     putchar('f');
     return 0;
```

```
}
```

Ответ: **acdbf** либо **adcbf**. Процесс-сын блокируется на первом вызове read(), поэтому первым символом будет напечатанный процессом-отцом символ ‘a’. Отец ждет завершения сына, поэтому последним символом будет напечатанный отцом ‘f’. После первой записи в канал fd отцом, сын печатает ‘d’, после чего сын заблокируется на read() из второго канала. Отец в это же время печатает ‘c’, так что порядок d и c может быть произвольным. После записи отцом в канал fd2, сын напечатает ‘b’.

9. Пусть в операционной системе используется файловая система, использующая FAT. Для представления номера блока в системе используется беззнаковое целое. Описать функцию, которая по номеру начального блока файла определяет размер файла в блоках. Функция принимает в качестве параметров номер начального блока файла и указатель на область памяти, в которой находится FAT.

Ответ: i-ая строка таблицы FAT хранит информацию о состоянии i-ого блока файловой системы, а также в ней указывается номер следующего блока файла. Для получения списка блоков файловой системы, в которых хранится содержимое конкретного файла, необходимо найти номер начального блока, а затем, последовательно обращаясь к таблице размещения и извлекая из каждой записи номер следующего блока, дойти до ссылки на нулевую строку таблицы. Нулевая строка таблицы уже не относится к рассматриваемому файлу (нулевой блок служебный и номер 0 не может передаваться в функцию в качестве номера начального блока.

```
int calculateSize(unsigned num, unsigned *fat)
{
    int counter = 0;
    while (num != 0) {
        num = fat[num];
        counter++;
    }
    return counter;
}
```

10. В компьютере используется 8-сегментная модель организации памяти. Описать функцию с заголовком `unsigned VirtIntoPhis(segment* SegTable, unsigned VirtAddr)`, преобразующую виртуальный сегментный адрес (заданный параметром `VirtAddr`) некоторого процесса в физический (возвращается как результат функции).
Все сегменты процесса размещены в физической памяти. Обнаружение ошибки преобразования виртуального адреса в физический должно приводить к завершению программы с кодом 25.
Таблица сегментов реализуется в виде массива структур типа `segment` и передается в программу по указателю `SegTable`.
Описать и прокомментировать все необходимые структуры данных при условии, что физические и виртуальные адреса имеют размер 32 бита (равный размеру типа `int`).

Ответ: Виртуальный адрес содержит номер сегмента и смещение в сегменте. Ввиду использования 8-сегментной организации памяти под номер сегмента отводится три первых бита в виртуальном адресе, а под смещение (offset) – оставшиеся 29 бит.
Номер строки таблицы сегментов – это номер сегмента. Содержимое строки таблицы сегментов – это размера сегмента (size) и адрес начала сегмента (SegAddr).

Физический адрес вычисляется как сумма адреса начала сегмента SegAddr и смещения offset. Прерывание происходит, если смещение больше либо равно размеру сегмента.

```
typedef struct Seg
    unsigned size;      /* размер сегмента */
    unsigned SegAddr;   /* физ. адрес начала сегмента */
} segment;

enum {SegErr=25}; /* код ошибки сегментации */

unsigned VirtIntoPhis(segment* SegTable, unsigned VirtAddr){

    unsigned SegNum = VirtAddr >> 29; /* получаем номер сегмента */
    unsigned offset = VirtAddr &0x1fffffff;/*смещение в сегменте*/
    if (SegTable[SegNum].size > offset)
         return SegTable[SegNum].SegAddr + offset;
    exit(SegErr);
}
```

Вариант 2

1. В ОЗУ 16-разрядного компьютера используется контроль целостности данных по четности. Описать возможную структуру ячейки памяти и ее побитовое содержимое для случая хранения в машинном слове четверичного числа 1033231_4.

Ответ: в 16-разрядном компьютере ячейка памяти содержит 16 бит данных, бит паритета (тег) – дополнительный, он в данные не входит.
Структура ячейки памяти: **0001 0011 1110 1101** – это данные, и еще один бит (приписывается справа или слева) равен **1** – он вычисляется как сумма по модулю 2 (`xor`) всех битов данных.

2. Пусть дано четверичное число 125432_8, являющееся адресом оперативной памяти, расслоенной по 8 банкам. Банку с каким номером принадлежит заданный адрес?

Ответ: **2.** В схеме расслоения памяти последовательные адреса размещаются в последовательных банках памяти. При 8 банках памяти за номер банка будут отвечать младшие 3 бита адреса. Они равны 010 (двоичное представление последней восьмеричной цифры), то есть 2 в десятичной системе. Банки нумеруются всегда с 0.

3. Пусть некоторый процесс с pid=F породил два сыновних процесса с pid=A и pid=B, реализованный программой на Си:

```
int  main (int argc, char **argv) /* pid=F */
{
    if (fork()==0){ /* pid=A */
        printf ("%d %d\n",getpid(),getppid());
        exit(0);
    }
    if (fork()==0){ /* pid=B */
        printf ("%d\n",getpid());
        exit(0);
    }
 return 0;
}
```

Считаем, что printf работает атомарно без буферизации и обращения ко всем системным вызовам срабатывают успешно, заголовочные файлы опущены. Перечислить все возможные комбинации значений, которые попадут на стандартное устройство вывода в результате выполнения данной программы.

Ответ: Процесс F не ждет процесс A (отсутствует wait()), поэтому F может завершиться как до завершения A (тогда осиротевший A станет потомком процесса init с pid=1), так и после. Никакой синхронизации между братьями A и B нет. Поэтому возможны четыре варианта ответа:

1	2	3	4
A F B	**A 1 B**	**B A F**	**B A 1**

4. Что будет выведено в стандартный поток вывода? Привести все возможные варианты ответа. Считаем, что обращение ко всем системным вызовам срабатывает успешно, а printf работает атомарно и без буферизации. 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;
}
```

Ответ: Процесс-отец ждет процесса-сына (есть вызов wait()), поэтому последней печатью будет “01” от отца. До этого момента в канал попадет ‘0’,’1’ либо ‘1’,’0’ из-за ситуации «гонок» при записи в канал, что будет прочитано и напечатано сыном.
Поэтому возможны два варианта ответа:

1	2
01	**10**
01	**01**

5. Дана файловая система, имеющая организацию в виде связного списка. Пусть ссылка в файловом блоке занимает 4 байта, а размер файлового блока равен 1024 байтам. Пусть некоторый файл из данной файловой системы содержит текстовую информацию (последовательность байтов, содержащих коды символов). За какое минимальное количество обменов можно прочесть часть текста с 3000 символа до 3050 символа (нумерация символов в тексте начинается с 1).

Ответ: Содержательная информация в файловом блоке занимает 1024 – 4 = 1020 байтов. При данной организации файловой системы придется прочитать последовательно все блоки (согласно списку) от первого до блока, содержащего 3050-й символ. Поделив 3050 на 1020, получим 2.99, то есть 3050-й (последний) символ находится в 3-м блоке (округляем 2.99 в большую сторону), то есть требуется выполнить **3 обмена**.

6. Пусть в 32-разрядном компьютере используется страничная память с двухуровневой таблицей страниц. Размер страницы 1024 байта. Каждая таблица второго уровня состоит из 512 записей. Сколько записей содержит таблица страниц первого уровня (внешняя)?

Ответ: Виртуальный адрес – это комбинация номера страницы первого уровня, номера страницы второго уровня и смещения в странице, всего 32 бита.
$512 = 2^9$, следовательно под номер страницы второго уровня отводится 9 бит.
$1024 = 2^{10}$, следовательно под смещение в странице отводится 10 бит.
На номер внешней страницы остается 32-9-10=13 бит, следовательно внешняя таблица содержит $2^{13} =$ **8192** записей (не байтов!).

7. Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран срабатывает атомарно и без буферизации. Все системные вызовы срабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
     pid_t pid;
     int fd[2];
     char x[] = "abc";
     pipe(fd);
     if( (pid = fork()) == 0 ) {
          write(1, x, 1); x[0]='f';
          write(fd[1], &x, 2); x[0]='s';
     }
     else {
          read(fd[0], &x, 2);
          kill(pid, SIGKILL);
          wait(NULL);
     }
     write(1, x, 1);
     return 0;
}
```

Ответ: **af** либо **asf**. Отцовский процесс блокируется на чтении из канала до тех пор, пока сын не напечатает ‘a’ (начальный элемент в x) и не запишет в канал последовательность ‘f’, ‘b’. Эту последовательность отец запишет в начало своей копии переменной x, пошлёт сыну сигнал и будет ждать его завершения, после чего напечатает ‘f’. Сын может успеть до получения сигнала (приводящего к его завершению) напечатать ‘s’, а может не успеть. Поэтому возможных вариантов ответа два.

8. Что будет выведено на экран? Если возможны несколько вариантов – привести все. Предполагается, что обращение к функции вывода на экран срабатывает атомарно и без буферизации. Все системные вызовы срабатывают успешно. Подключение заголовочных файлов опущено.

```
int main()
{
     char c='d';
     int fd[2], fd2[2];
     pipe(fd);
     pipe(fd2);
     if(fork() == 0) {
          write(fd[1], &c,1); printf("3");
          read(fd2[0], &c,1); printf("1");
          exit(0);
     }
     printf("0");
     read(fd[0], &c, 1);
     printf("2");
     write(fd2[1], &c, 1);
     wait(NULL);
     printf("4");
```

```
    return 0;
}
```

Ответ: **02314** либо **03214** либо **30214**. Отец ждет завершения сына, поэтому последним символом будет напечатанный отцом символ ‘4’. Перед завершением сын ждет символ в канале fd2, получив который печатает ‘1’. К этому моменту отец уже напечатал символы ‘0’ и ‘2’, так что предпоследним будет символ ‘1’. Символ ‘3’ печатается сыном параллельно с отцовской последовательностью ‘0’, ‘2’ и может «вклиниться» в любое место этой последовательности: 02**3**, 0**3**2, **3**02. Поэтому есть три возможных ответа.

9. Пусть в операционной системе используется файловая система, использующая FAT. Для представления номера блока в системе используется беззнаковое целое. Описать функцию, которая по номеру начального блока файла определяет размер файла в блоках. Функция принимает в качестве параметров номер начального блока файла и указатель на область памяти, в которой находится FAT.

Ответ: i-ая строка таблицы FAT хранит информацию о состоянии i-ого блока файловой системы, а также в ней указывается номер следующего блока файла. Для получения списка блоков файловой системы, в которых хранится содержимое конкретного файла, необходимо найти номер начального блока, а затем, последовательно обращаясь к таблице размещения и извлекая из каждой записи номер следующего блока, дойти до ссылки на нулевую строку таблицы. Нулевая строка таблицы уже не относится к рассматриваемому файлу (нулевой блок служебный и номер 0 не может передаваться в функцию в качестве номера начального блока.

```
int calculateSize(unsigned num, unsigned *fat)
{
    int counter = 0;
    while (num != 0) {
        num = fat[num];
        counter++;
    }
    return counter;
}
```

10. Описать на языке Си функцию преобразования 32-разрядного виртуального адреса в 32-разрядный физический адрес с использованием прямой таблицы страниц. Функция должна иметь следующий прототип:
`unsigned VA_to_PA(unsigned Virt_A, unsigned *Page_Tab)`, где Virt_A – виртуальный адрес, Page_Tab – таблица страниц. Размер страницы – 1024 байта. Считаем, что все виртуальные страницы данного процесса размещены в ОЗУ. Каждый элемент таблицы занимает 32 бита. В решении должно быть представлено описание структуры таблицы страниц и интерпретация ее содержимого.

Ответ: Предположим, что размер типа объекта типа unsigned – 32 бита. Младшие 10 битов виртуального адреса – смещение, старшие 22 бита – номер страницы. Предположим, что таблица страниц одноуровневая и номер страницы располагается в младших 22 битах элемента таблицы страниц.

```
unsigned VA_to_PA(unsigned Virt_A, unsigned *Page_Tab) {
    unsigned offset = Virt_A & ((1 << 10) - 1);
    unsigned vpn = Virt_A  >> 10;
    unsigned ppn = Page_Tab [vpn] & ((1 << 22) - 1);
```

```
    return (ppn << 10) | offset;
}
```