Коллоквиум ОС 2017

1. Описать на Си функцию To_PA, обеспечивающую перевод страничного 32-х разрядного адреса в 32-х разрядный физический. Функция должна иметь следующий прототип:
   unsigned To_PA(unsigned * Page_Tab, unsigned Virt_A, int PID);

   где Page_Tab – таблица страниц; Virt_A – виртуальный адрес; PID – идентификатор «контролирующего» процесса. Считаем, что размер страницы равен 512 машинных слов, минимальной адресуемой единицей памяти является машинное слово, размер переменной типа unsigned – 32 бита (машинное слово). Функция возвращает сформированный физический адрес, если соответствующая виртуальная страница находится в памяти. Если виртуальной страницы нет в памяти, то функция отправляет «контролирующему» процессу сигнал SIGPL (предопределенная вне функции константа). В решении должно быть представлено описание структуры таблицы страниц и интерпретация ее содержимого. (Вес задачи: 3)
2. В файловой системе используются битовые массивы для хранения информации о свободных и занятых блоках. Написать на Си функцию, принимающую в качестве параметров указатель на начало этого битового массива (последовательность байтов), номер блока файловой системы, максимально возможный номер блока и возвращающую статус занятости этого блока: 0 – свободен, 1 – занят, -1 – номер вне диапазона. (Вес: 1)
3. Написать программу на Си, которая свяжет неименованным каналом два процесса, являющиеся внуками исходного процесса. Один из внуков передает другому PID своего родителя и свой PID и завершается, а другой выводит на стандартный вывод полученную информацию и также завершается. (Вес: 1)
4. Сколько раз система обратится к содержимому индексных дескрипторов при вызове:
   open(”/dir/dir/file”, O_RDONLY) ? Обосновать ответ. Считаем, что ни один из элементов пути к файлу не является символической ссылкой. (Вес: 1)

Ответы к коллоквиуму 2017 по ОС

```
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <stdio.h>

/* Задача 1 */

enum {SIGPL, No_Page};
/* Page_Tab - это массив из беззнаковых целых */
/* unsigned = 23 бита для FP | флаги | младший бит присутствия */

unsigned To_PA(unsigned *Page_Tab, unsigned Virt_A, int PID){
    unsigned VP = Virt_A>>9;
    unsigned Offset = Virt_A & 0xFFu;
    int present = Page_Tab[VP] & 1;
    if (present) {
        unsigned FP = Page_Tab[VP] >> 9;
        return FP << 9 | Offset;
    } else {
        kill(PID, SIGPL);
        _exit(No_Page);
    }
}



/* Задача 2 */

/* Полагаем, что CHAR_BIT == 8, блоки нумеруются от 0 до Max_Num */

int is_free (unsigned char *BitBlocks, unsigned Num, unsigned Max_Num)
{
    if ( Num >= Max_Num) return -1;
    else return (BitBlocks[Num >> 3]) >> (7 - (Num & 7u))) & 1;
}
```

```
/* Задача 3 */

/* полагаем, что pid_t совпадает с  int */

int main(void) {
    if (fork()==0){
        int fd[2];
        pipe(fd);
        if (fork()==0) {
            int pid[] = {getppid(), getpid()};
            write(fd[1], &pid, sizeof pid);
        } else if (fork()==0) {
            int pid[2];
            read(fd[0],&pid, sizeof pid);
            printf("%d %d\n", pid[0], pid[1]);
        }
     }
}

/* Задача 4 */

/* 1 – inod для корня /, чтобы найти inod для подкаталога "dir"
 * 2 – inod для /dir,  чтобы найти inod для подкаталога "dir"
 * 3 – inod для /dir/dir, чтобы найти inod для файла "file"
 * 4 – inod для /dir/dir/file, чтобы проверить права доступа и
загрузить файл в память
Ответ: 4
*/
```