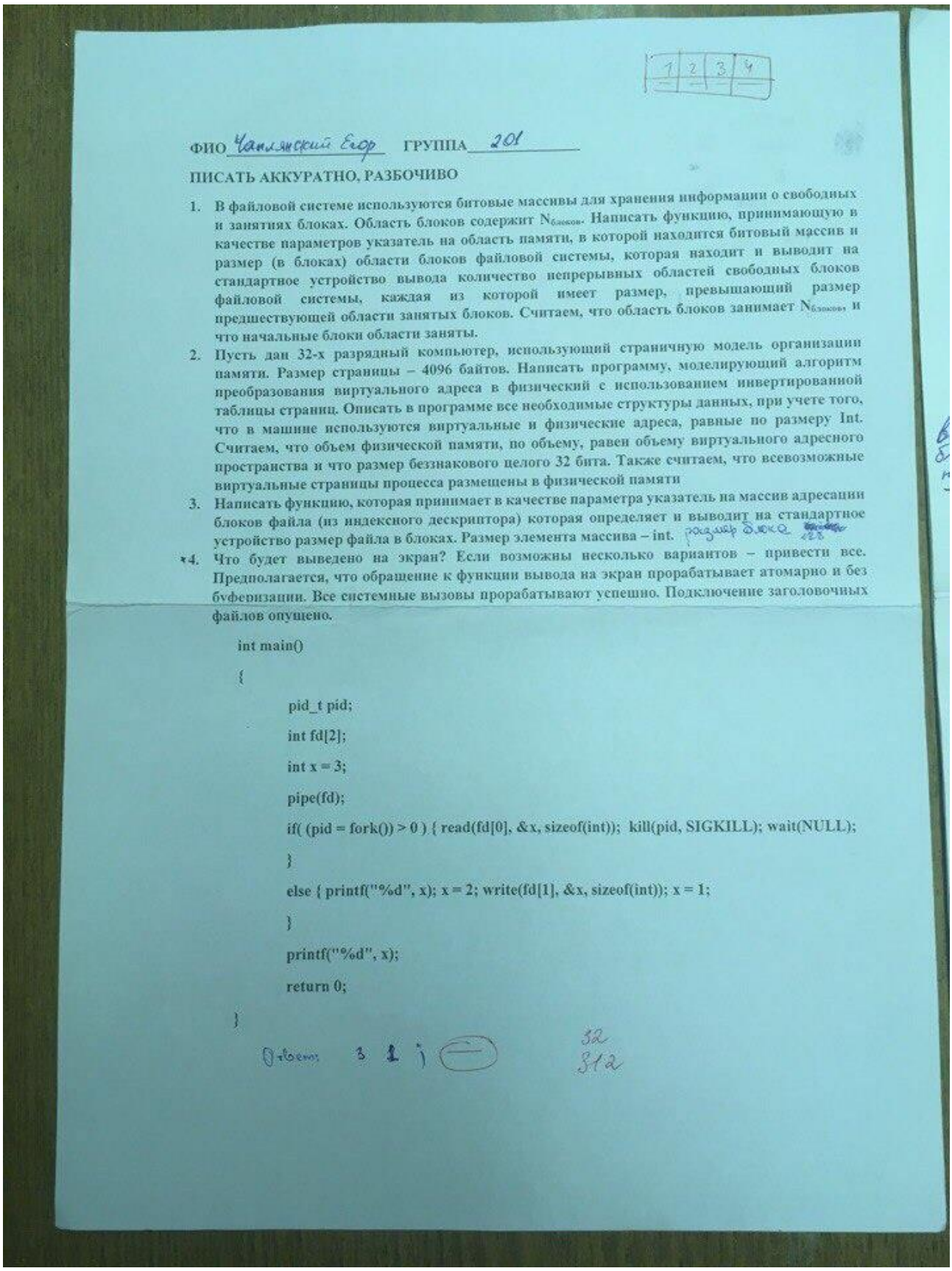


ФИО Чапланский Егор ГРУППА 201

ПИСАТЬ АККУРАТНО, РАЗБОЧИВО

1. В файловой системе используются битовые массивы для хранения информации о свободных и занятиях блоках. Область блоков содержит $N_{блоков}$. Написать функцию, принимающую в качестве параметров указатель на область памяти, в которой находится битовый массив и размер (в блоках) области блоков файловой системы, которая находит и выводит на стандартное устройство вывода количество непрерывных областей свободных блоков файловой системы, каждая из которой имеет размер, превышающий размер предшествующей области занятых блоков. Считаем, что область блоков занимает $N_{блоков}$ и что начальные блоки области заняты.
2. Пусть дан 32-х разрядный компьютер, использующий страничную модель организации памяти. Размер страницы – 4096 байтов. Написать программу, моделирующий алгоритм преобразования виртуального адреса в физический с использованием инвертированной таблицы страниц. Описать в программе все необходимые структуры данных, при учете того, что в машине используются виртуальные и физические адреса, равные по размеру Int. Считаем, что объем физической памяти, по объему, равен объему виртуального адресного пространства и что размер беззнакового целого 32 бита. Также считаем, что всевозможные виртуальные страницы процесса размещены в физической памяти
3. Написать функцию, которая принимает в качестве параметра указатель на массив адресации блоков файла (из индексного дескриптора) которая определяет и выводит на стандартное устройство размер файла в блоках. Размер элемента массива – int.
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()
{
        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 = 2; write(fd[1], &x, sizeof(int)); x = 1;
        }
        printf("%d", x);
        return 0;
}
```

Номер 1.
Не разбирали на семинаре.

Номер 2. Так как размер страницы 4096, то на смещение по странице уходит 12 бит. Значит, на количество страниц уходит 32 — 12 = 20 бит.

У нас есть таблица вида

pid	Виртуальный адрес (20 бит)

Алгоритм действий:
1. берем va (виртуальный адрес), первые 20 бит — номер вирт.страницы, последние 12 — смещение.
2.берем pid нашего процесса
3.Ищем по таблице строку, в которой pid и виртуальный адрес совпадают.
Номер этой строки — номер физической страницы.
4. Физический адрес: первые 20 бит — номер физ.страницы, последние 12 бит — смещение(см 1 пункт)

```
struct Elem {   // структура элемент таблицы
        pid_t pid;
        unsigned virt_addr;
};

struct Elem itable[1u<<20];  //  массив из 2 ^20 структур, 1u<<20 — сдвигаем беззнаковую 1 на 20

unsigned f(unsigned va) {
        unsigned offset = va & 0xfff; // последние 12 бит — смещение
        unsigned begin = va > > 12; // первые 20 бит — номер вирт.страницы
        pid_t p = getpid();
        for (int i = 0; i < 32; i++) {
                if (itable[i].pid == p && itable[i].virt_addr == begin) {
                        return i < < 12 | offset; // первые 20 бит — номер физ.стр., последние 12 - смещение
                }
        }
        kill(p, SIGUSR1); если нет в таблице посылаем сами себе сигнал ( мы так на семинаре делали(возможно, не обязательно)
        return 0;
}
```

Номер 3.

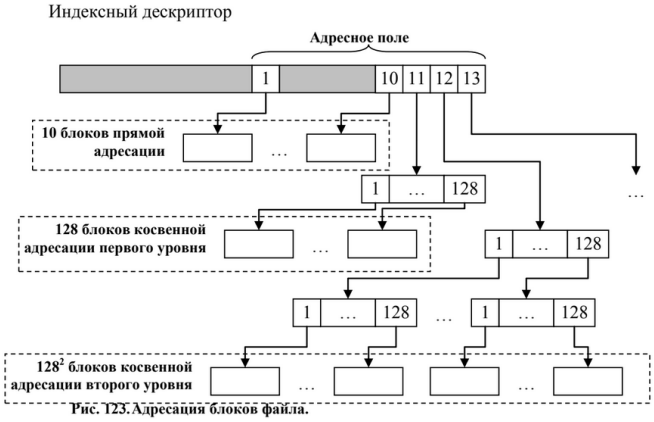


Рис. 123. Адресация блоков файла.

Размер не д.б. 128 (это пример из машбука).

Если встретили 0, значит, файл закончился.

```
unsigned f (unsigned *point, unsigned block_size){
	unsigned ans = 0;
	for (int i = 0; i < 10; i++) {
		if (point[i] == 0) {
			return ans;
		}
		ans++;
	}
	unsinged count_elem = block_size / sizeof(int);
	for (int i = 0; i < count_elem; i++) {
		if (point[10][i] == 0) {
			return ans;
		}
		ans++;
	}
	for (int i = 0; i < count_elem; i++) {
		for (int j = 0; j < count_elem; j++) {
			if (point[11][i][j] = 0) {
				return ans;
			}
			ans++;
		}
	}
	for (int i = 0; i < count_elem; i++) {
		for (int j = 0; j < count_elem; j++) {
			for (int k = 0; k < count_elem; k++) {
				if (point[12][i][j][k] = 0) {
					return ans;
				}
				ans++;
			}
		}
	}
	return ans;// если все занято
}
```

Номер 4.
См. картинку

1	2	3	4
+	–	–	+

ФИО Марцев Глеб Олегович ГРУППА 202

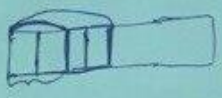
ПИСАТЬ АККУРАТНО, РАЗБОЧИВО

1. **В файловой системе используются битовые массивы для хранения информации о свободных и занятиях блоках. Область блоков содержит $N_{блоков}$. Написать функцию, принимающую в качестве параметров указатель на область памяти, в которой находится битовый массив и размер (в блоках) области блоков файловой системы, которая находит выводит на стандартное устройство вывода размер самой большой непрерывной области свободных блоков файловой системы.**
2. **В компьютере используется 4-х сегментная модель организации памяти. Написать программу, реализующую преобразование виртуального сегментного адреса в физический (считаем, что используются все 4 сегмента). Ситуацию прерывания реализовать отправкой процессом сигнала SigUsr самому себе. Описать в программе все необходимые структуры данных, при учете того, что в машине используются виртуальные и физические адреса, равные по размеру Int.**
3. **Написать функцию, которая принимает в качестве параметра указатель на массив адресации блоков файла (из индексного дескриптора) которая выводит на стандартное устройство вывода найденный номер блока файла, имеющий минимальное значение. Размер элемента массива – int.**

вывести → блок с мин номером

N - разм блока, перед ф-ию !

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. Пусть msgId – идентификатор существующей пустой очереди сообщений.**



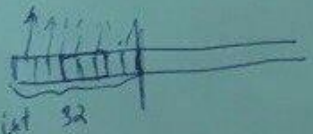
```
struct
{
 long type;
 char data[1];
} msg;
.....
msg.type = 1; msg.data[0] = 'a'; msgsnd(msgId, &msg, 1, 0);
msg.type = 2; msg.data[0] = 'b'; msgsnd(msgId, &msg, 1, 0);
msg.type = 2; msg.data[0] = 'c'; msgsnd(msgId, &msg, 1, 0);
msg.type = 1; msg.data[0] = 'd'; msgsnd(msgId, &msg, 1, 0);
(I) msgrcv(msgId, &msg, 1, 2, 0); putchar(msg.data[0]);
(II) msgrcv(msgId, &msg, 1, 0, 0); putchar(msg.data[0]);
(III) msgrcv(msgId, &msg, 1, 1, 0); putchar(msg.data[0]);
.....
```

main

autosel

a->b->c->d
① a->d
② b->c
(I) → a->c->d
(II) ① a->d ② c
c->d
(III) ① d ② c

Ответ: b a d



Номер 1.
На семинаре считали, что у нас little-endian. Поэтому,если бы у насв «числе» было бы по 4 бита, то в примере 6 15 3 0
1010 **1111** 00**11** 0000 — запись чисел, 010**1 1111 11**00 0000 — в little-endian
Ответ был бы 7. Поэтому мы идем справа налево.

Считаем, что N — количество блоков, делится на 32.(Т.е. весь битовый массив заполнен).

```
void f(const unsigned *a, int N) {
	int i, j = 0, k, max = 0;
	unsigned int n = N / 32;
	for (i = 0; i < n; i++) {
		unsigned int b = a[i];
		for (j = 0; j < 32; j++) {
			if (b & 1) {
				k++;
			} else {
				if (max < k) {
					max = k;
				}
				k = 0;
			}
			b >>= 1;
		}
	}
	if (max < k) {
		max = k;
	}
	printf(«%d\n», max);
}
```

Номер 2.
Так как 4 сегмента, то первые 2 бита из адреса — на хранение номера сегмента, 30 бит — на смещение

```
struct Seg {   // элемент таблицы
	unsigned size;
	unsigned phys_addr;
};

struct Seg table[4]; //таблица из 4 сегментов

unsigned f(unsigned virt_addr) {
	int n = virt_addr > > 30; // номер сегмента
	int offset = virt_addr & 0x3fff ffff;   //смещение
	if (offset >= size) {  // если необходимое смещение в физ.странице больше чем ее размер
		kill(getpid(), SIGUSR1);
		return 0;
	}
	return table[n].phys_addr + offset;
}
```

Номер 3.
Не разбирали на семинаре

Номер 4.
См. картинку.ы