ВМК МГУ - пробный коллоквиум по ОС-1

RAM-1

В оперативной памяти модельного 32-битного Big Endian-компьютера используется индикатор состояния ячейки памяти (готова-не готова). Помимо этого в ячейке используется проверка корректности состояния данных и значения готовности страницы по чётности, где за каждые 3 бита данных, начиная с младшего бита, отвечает 1 бит и ещё один бит отвечает за корректность состояния готовности страницы.
Опишите структуру ячейки памяти и заполните её для:
А. Страница готова, содержимое: $(\texttt{0xDEADBEEF})_{16}$
Б. Страница не готова, содержимое: массив `['\0', '\0', 45, 55]`

WWW-1

Происходит передача пакета между двумя адресами стандарта IPv4. Определите типы каждого из адресов, адреса каждой из сетей и каждого из хостов (если это возможно). Укажите, через какие уровни TCP/IP будет проходить пакет.
А. `192.168.0.1` → `10.0.0.25`
Б. `124.234.256.109` → `0.0.0.0`

WWW-2

Условие аналогично задаче WWW-1, но теперь коммутация пакета происходит через шлюз.
А. `192.168.0.128` → `178.16.16.1` → `192.168.64.64`
Б. `10.0.0.255` → `128.16.16.1` → `192.168.122.55`

PIPE-1

Со стандартного потока ввода считывается число n из интервала (0, 64). Также считывается два числа типа `uint64_t` - маски. Создается n анонимных каналов, далее создается n процессов - процесс с номером i берет i-й бит числа x, в случае равенства его 1 он записывает в канал с номером i свой PID. В противном случае в канал записывается -PID. После отработки всех процессов отец считывает числа из каналов последовательно, создав переменную-аккумулятор. Если i-й бит числа y установлен, число из канала с номером i должно быть прибавлено к аккумулятору, в противном случае число игнорируется. Вывести значение аккумулятора после вышеописанных действий.

C-1

Напишите все варианты вывода программы:

```
int
main(void)
{
        int fd[2]; pipe(fd);
        int pid = 0;
        if (!(pid = fork())) {
                int n = 0;
                read(fd[0], &n, sizeof(n));
                printf("%d␣", n);
                n = 00024;
                write(fd[1], &n, sizeof(n));
        } else if (!pid) {
                read(fd[0], &n, sizeof(n));
                write(fd[1], &n, sizeof(n));
                printf("%d␣", n);
        }

        int a = 0xF;
        write(fd[1], &a, sizeof(a));

        if (!pid) {
                read(fd[0], &a, sizeof(a));
        }
        printf("%d␣", a);

        if (pid) {
                goto exit;
        }

        int status = 0;
        wait(&status);
        if (WIFEXITED(status)) {
                printf("%d\n",
                WEXITSTATUS(status));
                goto exit;
        }
        printf("\n");

exit:
        return 0;
}
```

INODE-1

Опишите структуру индексного дескриптора SysVFS.

INODE-2

В файловой системе SysVFS хранится текстовый файл без метаданных. Символы закодированы широкой ASCII. Пользуясь ответом предыдущей задачи, напишите, сколько обменов потребуется для чтения 2023-го символа начиная с открытия файла и заканчивая получением символа. Размер блока считайте равным 512 байтам.

THREAD-1

Пусть есть глобальная переменная x. x имеет тип static int и равен изначально нулю. Предложить реализацию функции-носителя для двух нитей, играющих в пинг-понг переменной x (классическая формулировка). Останова не требуется, в аргументе можете передавать любую необходимую информацию.

STAT-1

Пусть процесс с PID X порождает сына Y, а тот порождает внука Z. Найти математическое ожидание количества единиц на стандартном потоке вывода. Считаем распределение конфигураций планировщика равномерным. Считаем, что PPID(X) == 1. Буферизацией и неуспехами системных вызовов пренебречь.

```
int
main(int argc, char *argv[])
{
    if (!fork()) {
        fork();
        printf("%d\n", getppid());
    }
    printf("%d␣%d\n", getpid(), getppid());
    printf("%d\n", getppid());
}
```

ВМК МГУ - пробный коллоквиум по ОС-2

THREAD-0

Приведите пример числа, подаваемого на поток ввода для которого вывод программы может содержать ровно 35 единиц подряд.

```
typedef union
{
    unsigned char a[8];
    unsigned long long d;
} myunion;

pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
int p[2];
myunion u;

void
*work(void *arg)
{
    int serial = *(int *) arg;
    pthread_mutex_lock(&m);
    write(p[1], u.a + serial, sizeof(u.a[0]));
    pthread_mutex_unlock(&m);
    return NULL;
}

int
main(void)
{
    pthread_t tids[CHAR_BIT];
    unsigned long long x;
    scanf("%llu", &x);
    u.d = x;
    pipe(p);
    for (int i = 0; i < CHAR_BIT; i++) {
        pthread_create(tids + i, NULL, work, &i);
    }
    for (int i = 0; i < CHAR_BIT; i++) {
        pthread_join(tids + i, NULL);
    }
    close(p[1]);
    read(p[0], &x, sizeof(x));
    printf("%b\n", x);
}
```

C-1

Напишите все варианты вывода программы:

```
int
main(void)
{
        int fd[2]; pipe(fd);
        int pid = 0;
        if (!(pid = fork())) {
                int n = 0;
                read(fd[0], &n, sizeof(n));
                printf("%d ", n);
                n = 00024;
                write(fd[1], &n, sizeof(n));
        } else if (!pid) {
                read(fd[0], &n, sizeof(n));
                write(fd[1], &n, sizeof(n));
                printf("%d ", n);
        }

        int a = 0xF;
        write(fd[1], &a, sizeof(a));

        if (!pid) {
                read(fd[0], &a, sizeof(a));
        }
        printf("%d ", a);

        if (pid) {
                goto exit;
        }

        int status = 0;
        wait(&status);
        if (WIFEXITED(status)) {
                printf("%d\n",
                WEXITSTATUS(status));
                goto exit;
        }
        printf("\n");

exit:
        return 0;
}
```

C-2

Напишите все варианты вывода программы:

```
int
main(void)
{
        char n = 10;
        int fd[2]; pipe(fd);
        pid_t pid = fork();
        if (!pid) {
                read(fd[0], &n, sizeof(n));
                printf("%d ", n);
                n = -1;
                write(fd[1], &n, sizeof(n));
                n = 4;
                write(fd[1], &n, sizeof(n));
                read(fd[0], &n, sizeof(n));
                printf("%d\n", n);
        } else {
                write(fd[1], &n, sizeof(n));
                printf("%d %u\n", getpid(), n);
                read(fd[0], &n, sizeof(n));
                printf("%d\n", n);
                read(fd[0], &n, sizeof(n));
        }

        n = 24;
        write(fd[1], &n, sizeof(n));

        printf("%d %d", getpid(), n);

        wait(NULL);

        printf("%d %u", getpid(), n);

        return 0;
}
```