Контрольная работа №2.

Вариант 2. **ФИО:** Холопкин, ~~111~~ 211

Задание 1. Написать программу.

Напишите программу, которая будет ждать получения сигналов. Причём на один из сигналов (например, SIGINT) программа должна написать в stderr: «Отстань». Получение другого типа сигналов программа должна игнорировать. А на третий тип сигналов она должна написать в stdout: «Согласна», - и закончить своё выполнение.

Замечания: Можно использовать функции ввода/вывода как высокого, так и низкого уровней. Механизм окончания выполнения программы – произвольный. Тип сигналов первого, второго и третьего вида – на ваш выбор.

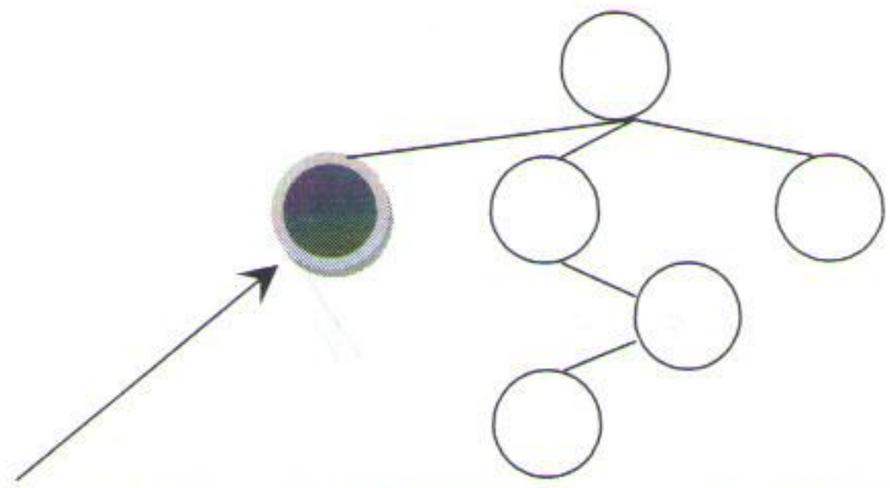
Задание 2. Написать программу.

На стандартный вход программе подаётся имя текстового файла. Программа должна вывести в стандартный вывод все строки файла, не содержащие подстроку «Copyleft». Строки, содержащие подстроку «Copyleft», должны выводиться в стандартный поток ошибок.

Замечания: Использовать функции чтения/записи только низкого уровня. Слово «copyleft» считать отличным от «Copyleft».

Задание 3. Написать программу.

После запуска программы, она должна породить следующее «генеалогическое» древо процессов:



- после чего этот процесс должен выполнить код из выполнимого файла "/usr/bin/root".

Замечание: Последовательность рождения процессов на ваше усмотрение.

Задание 4. Написать программу.

В программу через командную строку подаются дав слова двух файлов. Программа должна вывести на экран слово «Same», если слова одинаковы, или «Different», если различны.

Например:

«./a.out Win win» должно выдать «Different»/

«./a.out Win Win» должно выдать «Same»/

Задание 1.

```
void reaction1(int sig)
{
    fprintf(stderr, "Остань");
    return;
}

void reaction3(int sig)
{
    fprintf(stdout, "сопасна");
    exit(0);
}

int main()
{
    signal(SIGINT, reaction1);
    signal(SIGUSR1, SIG_IGN);
    signal(SIGCHLD, reaction3);
    while(1);
    return 0;
}
```

+

Задача 2.

```
int search(const char *s1, const char *s2) // ф-ция поиска s2 в s1.
{
    int i=0;
    int j=0;
    while (s1[i] != '\0')
    {
        while((s1[i+j] == s2[j]) && (s2[j] != '\0'))
            j++;
        if (s2[j] == '\0')
            return 1;
        else
            j=0;
        i++;
    }
    return 0;
}
#define SIZE 1000
int main()
{
    int fd;
    int buf = -2;
```

ФИО: Хохлачев, ИУ

Задача 2 (продолжение)

```
int i=0;
int k=10;
char filename[SIZE];
char str[SIZE]; — ограничение длины строки файла
while (buf != '\n')
{
    read(0, &buf, 1);
    filename[i] = (char) buf;
    i++;
}
filename[i-1] = '\0';
fd = open(filename, O_RDONLY);
while (k>0)
{
    i=0;
    while (((k = read(fd, str[i], 1)) == 1) && (str[i] != '\n'))
        i++;
    str[i] = '\0';
    if (search(str, "copyleft"))
        write(2, str, i);
    else
        write(1, str, i);
}
return 0;
}
```

Задание 3.

```
int main()
{ if (fork()>0)
  { if (fork()==0)
    { if (fork()==0)
      { fork();
      }
    }
  }
}
else
{ execl("usr/bin/root", "root", NULL);
  printf("Critical error");
}
return 0;
```

Задание 4.

```
int equal (char *s1, char *s2)
{
    int i=0;
    int j=0;
    while ((s1[i] == s2[i]) && (s1[i] != '\0') && (s2[i] != '\0'))
        i++;
    if ((s1[i] == '\0') && (s2[i] == '\0'))
        return 1;
    else return 0;
}
int main (int argc, char *argv[])
{
    if (equal (argv[1], argv[2]))
        printf ("same");
    else
        printf ("different");
    return 0;
}
```

+