

# Практикум на ЭВМ

## Семестровая работа №1.

### bash

Баев А.Ж.

Казахстанский филиал МГУ

26 октября 2019

## Этапы.

Было:

1. Делим на лексемы.
2. Стандартный запуск программы.
3. Перенаправление ввода и вывода.
4. Конвейер для двух элементов.
5. Конвейер для произвольного количества.

Сегодня:

6. Фоновый режим.
7. Смена директории `cd`.
8. Конвейер `&&`.
9. `Ctrl + C`.

## Этап 6. Фоновый режим

```
firefox &
```

Не делаем wait в родителе.

```
if (fork() == 0) {  
    exec();  
}  
/* no wait(NULL) */
```

## Этап 7. Смена директории

```
cd sources/  
cd  
cd ~  
cd ..
```

```
char **cmd;  
char *home;  
char *parent;  
...  
if (strcmp(cmd[0], "cd") == 0) {  
    if (cmd[1] == NULL || strcmp(cmd[0], "~") == 0) {  
        chdir(home);  
    } else {  
        chdir(cmd[1]);  
    }  
}
```

# Переменные окружения

```
1 #include <stdlib.h>
2 char *getenv(const char *name);
3 int  setenv(const char *name,
4             const char *value,
5             int  overwrite);
```

HOME	Путь к домашнему каталогу текущего пользователя.
USER	Имя текущего пользователя
PATH	Список каталогов для поиска исполняемых программ.
PWD	Путь к текущему рабочему каталогу.
SHELL	Интерпретатор по умолчанию.

```
1     const char home = getenv("HOME");
```

## Этап 8. Конвейер &&

```
1  cp super.c prog.c && gcc prog.c -o prog && ./prog
```

Ждем окончания выполнения  $i$ -й программы.

```
1  #include <stdio.h>
2  #include <sys/types.h> /* wait, fork */
3  #include <sys/wait.h> /* wait */
4  #include <unistd.h> /* fork, exec */
5
6  int main() {
7      if (fork() == 0) {
8          execl("prog", "prog", NULL);
9      }
10     int wstatus;
11     wait(&wstatus);
12     printf("%d\n", WEXITSTATUS(wstatus));
13     return 0;
14 }
```

## Этап 9. Ctrl + C

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <unistd.h>
4
5 void handler(int signo) {
6     puts("received_\SIGINT");
7 }
8
9 int main(void) {
10     signal(SIGINT, handler);
11     sleep(60);
12     return 0;
13 }
```

# Сигналы

Системные вызовы — из пользовательского кода вызвать код ядра.

Сигналы — из кода ядра вызвать пользовательский код.

Стандартное средство (открываем еще один терминал).

```
1 kill -SIGKILL 12345
```

где SIGKILL — сигнал, 12345 — pid процесса, которому посылается сигнал.

Нестандартное средство.

```
1 htop
2 k
3 enter
```



## Этап 9. Ctrl + C

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <unistd.h>
4
5 void inf_loop() {
6     ...
7 }
8
9 void handler(int signo) {
10     ...kill all son process...
11     ...free all resources...
12     inf_loop();
13 }
14
15 int main(void) {
16     signal(SIGINT, handler);
17     inf_loop();
18     return 0;
19 }
```

## Этап 9. Ctrl + C

Завершить процесс можно системным вызовом

```
1      #include <sys/types.h>
2      #include <signal.h>
3
4      int kill(pid_t pid, int sig);
```

Подробности

```
1  man 2 kill
```