Практикум на ЭВМ
Семестровая работа №1.
bash

Баев А.Ж.

Казахстанский филиал МГУ

26 октября 2018

Этапы.

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

Перенаправление вывода дочки

Перенаправим вывод дочернего процесса на стандартный ввод родительского процесса.

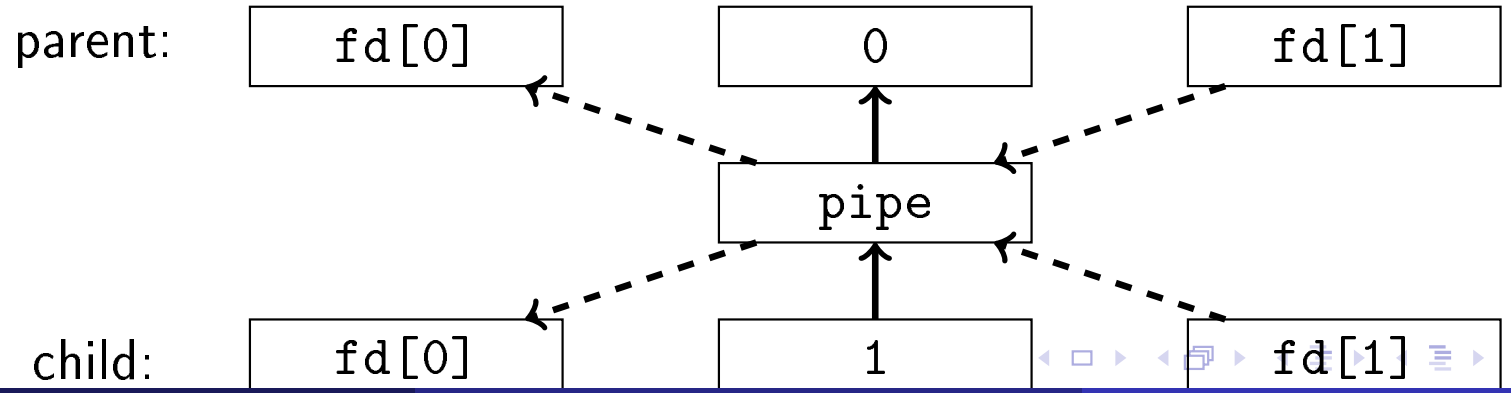
Этап 4. Именованный канал

```
#include <unistd.h>
int pipe(int pipefd[2]);
```

Именованный канал существует в системе и после завершения процесса.

Родитель получает сообщение от дочки

```
int fd[2];
pipe(fd);
if (fork() == 0) {
    close(fd[0]);
    dup2(fd[1], 1);
    close(fd[1]);
    execve(cmd[0], cmd)
    return 1;
}
close(fd[1]);
dup2(fd[0], 0);
close(fd[0]);
```

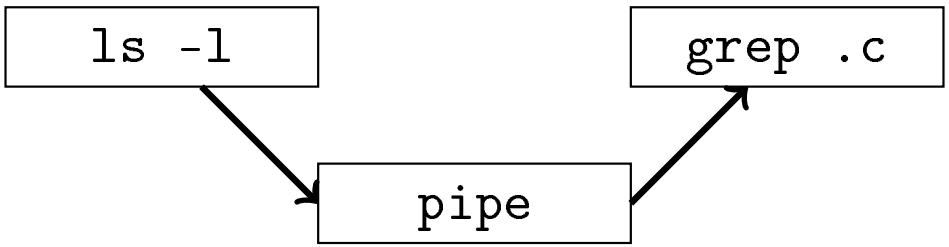


Двустороннее общение

```
int child2parent[2];
int parent2child[2];
pipe(child2parent);
pipe(parent2child);
if (fork() == 0) {
    close(child2parent[0]);
    dup2(child2parent[1], 1);
    close(parent2child[1]);
    dup2(parent2child[0], 0);
    execve(cmd[0], cmd)
    return 1;
}
close(child2parent[1]);
dup2(child2parent[0], 0);
close(parent2child[0]);
dup2(parent2child[1], 1);
```

Этап 4. Конвейер из двух программ.

```
ls -l | grep .c
```



Этап 4. Конвейер из двух программ.

```
char **cmd_A, **cmd_B;
int fd[2];
pipe(fd);
if (fork() == 0) {
    close(fd[0]);
    dup2(fd[1], 1);
    close(fd[1]);
    execve(cmd_A[0], cmd)
    return 1;
}
wait(NULL);
if (fork() == 0) {
    close(fd[1]);
    dup2(fd[0], 0);
    close(fd[0]);
    execve(cmd_B[0], cmd)
    return 1;
}
wait(NULL);
```

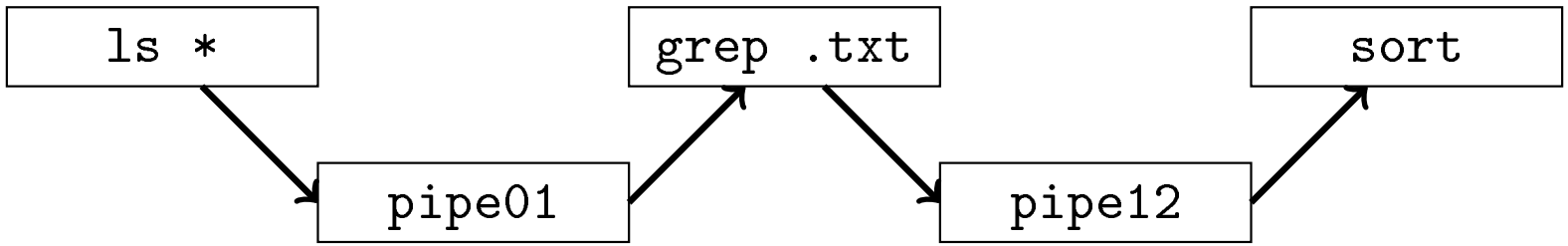
Этап 4. Конвейер из двух программ.

Должны продолжать работать перенаправления из файла и в файл.

```
ls * | sort > result.txt
```

Этап 5. Конвейер из n программ.

```
ls * | grep .txt | sort
```



Этап 5. Конвейер из n программ.

```
int (*fd)[2];
...
pipe(fd[i - 1]);
pipe(fd[i]);
...
if (fork() == 0) {
    close(fd[i - 1][1]);
    dup2(fd[i - 1][0], 0);
    close(fd[i - 1][0]);

    close(fd[i][0]);
    dup2(fd[i][1], 1);
    close(fd[i][1]);

    execve(cmd[i][0], cmd)
    return 1;
}
wait(NULL);
```

Тестируем динамическую память.

Открываем 2 терминала. Первый терминал

```
yes "true" | ./super_program
```

Команды: true, yes.

Второй терминал

```
top -d 1 -p $(pgrep super_program)
```

Команды: top, pgrep.