Практикум на ЭВМ
Семестровая работа №1.
bash

Баев А.Ж.

Казахстанский филиал МГУ

23 октября 2018

Пишем кастомный интерпретатор.

Было:
1. Делим на лексемы.
2. Стандартный запуск программы.
3. Перенаправление ввода и вывода.

Сегодня:
4. Конвейер для двух элементов.
5. Конвейер для произвольного количества.

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

Перенаправление вывода дочки

Перенаправим вывод дочернего процесса на стандартный ввод родительского процесса.

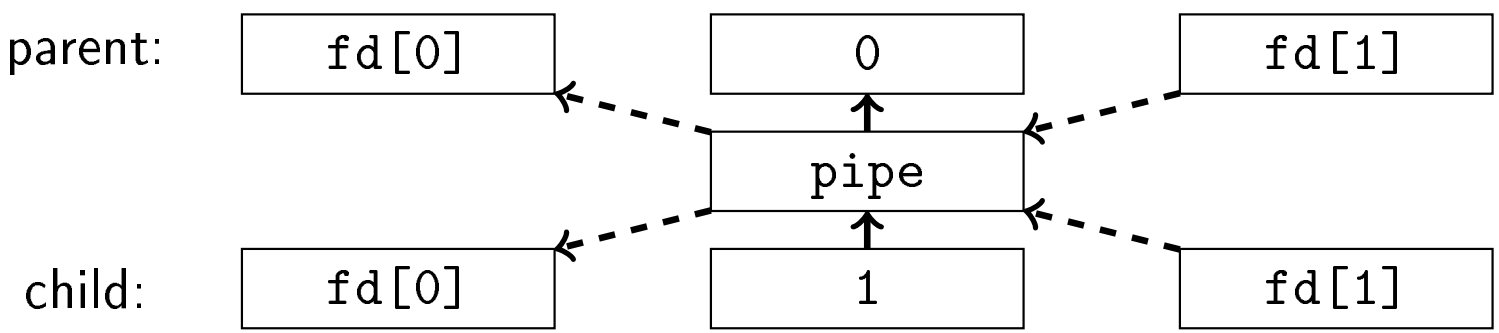
Этап 4. Именованный канал

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

Именованный канал существует в системе и после завершения процесса.

Родитель получает сообщение от дочки

```
 1  int fd[2];
 2  pipe(fd);
 3  if (fork() == 0) {
 4      dup2(fd[1], 1);
 5      close(fd[0]);
 6      close(fd[1]);
 7      execve(cmd[0], cmd)
 8      return 1;
 9  }
10  dup2(fd[0], 0);
11  close(fd[0]);
12  close(fd[1]);
```

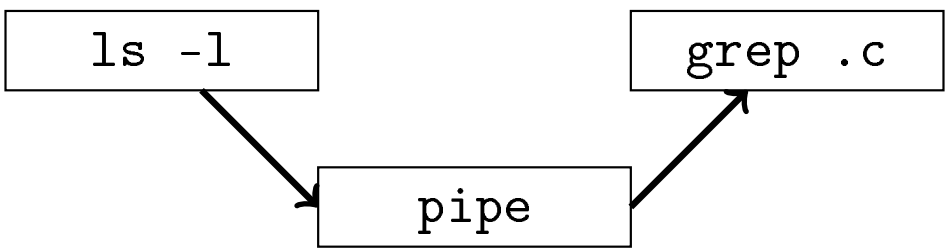


Двустороннее общение

```
 1 int child2parent[2];
 2 int parent2child[2];
 3 pipe(child2parent);
 4 pipe(parent2child);
 5 if (fork() == 0) {
 6     close(child2parent[0]);
 7     dup2(child2parent[1], 1);
 8     close(parent2child[1]);
 9     dup2(parent2child[0], 0);
10     execve(cmd[0], cmd)
11     return 1;
12 }
13 close(child2parent[1]);
14 dup2(child2parent[0], 0);
15 close(parent2child[0]);
16 dup2(parent2child[1], 1);
```

Этап 4. Конвейер из двух программ.

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



Этап 4. Конвейер из двух программ.

```
 1 char **cmd_A, **cmd_B;
 2 int fd[2];
 3 pipe(fd);
 4 if (fork() == 0) {
 5     dup2(fd[1], 1);
 6     close(fd[0]);
 7     close(fd[1]);
 8     execve(cmd_A[0], cmd)
 9     return 1;
10 }
11 wait(NULL);
12 if (fork() == 0) {
13     dup2(fd[0], 0);
14     close(fd[1]);
15     close(fd[0]);
16     execve(cmd_B[0], cmd)
17     return 1;
18 }
19 wait(NULL);
```

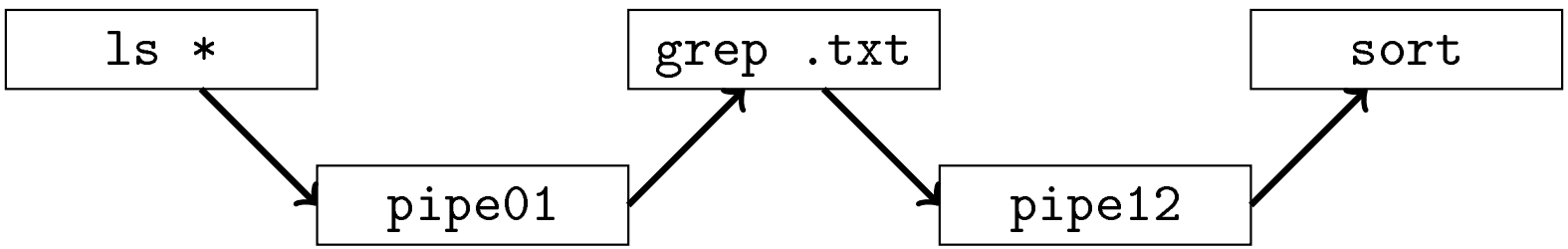
Этап 4. Конвейер из двух программ.

Должен работать с перенаправлением из файла (первой программы) и в файл (последней программы).

```
grep "include" < input.txt | sort > output.txt
```

Этап 5. Конвейер из *n* программ.

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



Этап 5. Конвейер из *n* программ.

```
 1 int (*fd)[2];
 2 ...
 3 pipe(fd[i - 1]);
 4 pipe(fd[i]);
 5 ...
 6 if (fork() == 0) {
 7     dup2(fd[i - 1][0], 0);
 8     close(fd[i - 1][1]);
 9     close(fd[i - 1][0]);
10
11     dup2(fd[i][1], 1);
12     close(fd[i][0]);
13     close(fd[i][1]);
14
15     execve(cmd[i][0], cmd)
16     return 1;
17 }
18 wait(NULL);
```

Код

```
1  while (...) {
2    int input_fd = 0, output_fd = 1;
3    char ***cmd_io_array = get_list(...);
4    char ***cmd_array = prepare_io(cmd_io_array,
5                                   &input_fd,
6                                   &output_fd);
7    int pipefd[pipes][2], pid;
8    for (int i = 0; i < pipes; ++i) {
9      pipe(pipefd[i + 1])
10     if ((pid = fork()) == 0) {
11       /* dup2 or close some fds*/
12       execvp(cmd_array[i][0], cmd_array[i]);
13     } else {
14       /* close some fds */
15       waitpid(pid, NULL, 0);
16     }
17   }
18   /* clear heap */
19 }
```

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.