MC504/MC514 - Sistemas Operacionais

Pipes

Islene Calciolari Garcia

Instituto de Computação - Unicamp

Segundo Semestre de 2016

Pipes

```
$ grep xxx log.txt > log-xxx.txt
$ wc -l log-xxx.txt
$ rm log-xxx.txt

$ grep xxx log.txt | wc -l
```

pipe()

```
int pipe (int FILEDES[2])
```

> *The 'pipe' function creates a pipe and puts the file descriptors for the reading and writing ends of the pipe (respectively) into 'FILEDES[0]' and 'FILEDES[1]'.*

Veja o código: `mypipe.c`

Pipe com entrada e saída padrão?

```
int dup2(int oldfd, int newfd);
```

dup2 makes newfd be the copy of oldfd, closing newfd first if necessary. After successful return of dup or dup2, the old and new descriptors may be used interchangeably.

Veja o código: `mypipe2.c`

Processos conectados de maneira transparente

```
$ cm1 <args1> | cmd2 <args2>
```

- A modificação da entrada e saída padrão deve ser feita antes da chamada a execve().
- Veja o código: minishell.c

popen()

```
FILE *popen(const char *command,
            const char *type);
int pclose(FILE *stream);
```

The popen() function opens a process by creating a pipe, forking, and invoking the shell. Since a pipe is by definition unidirectional, the type argument may specify only reading or writing, not both; the resulting stream is correspondingly read-only or write-only.

Veja o código: `mypopen.c` e `mypopen2.c`

Sincronismo entre os processos

O que acontece quando o processo que lê fica lento ou bloqueado?

- Faça o teste com o comando `yes`
  (com sorte, após algumas tentativas você verá `R (running)`)

```
$ ps -ef | grep yes
$ more /proc/XXXXXX/status
```

- Faça um teste com `yes | more`
  (se parar o `more`, o status de `yes` será sempre `S (sleeping)`)

```
$ ps -ef | grep more
$ more /proc/XXXXXX/status
$ ps -ef | grep yes
$ more /proc/XXXXXX/status
```