

Trabalho Prático 1 - Sistemas Operacionais

Matheus Filipe Sieiro Vargas

Setembro 2020

1 Shell Básico

1.1 Implementação

A primeira parte do trabalho prático consiste em explorar os conceitos básicos de chamadas do sistema para execução de comandos *shell*. Para tanto, foi disponibilizado o código esqueleto que implementa as estruturas necessárias para permitir que as seguintes funções fossem implementadas.

1.1.1 Comandos simples

A primeira tarefa consistia em implementar o caso de comandos simples e redirecionar estes comandos para as chamadas adequadas de sistema.

1.1.2 Redirecionamento

A segunda tarefa exigiu o conhecimento do fluxo de execução dos processos e a chamada de processos filhos.

1.1.3 Implementação de pipes

A terceira parte da implementação do *shell* consiste em prover um fluxo de execução provendo a saída de um comando como entrada para o próximo comando.

1.1.4 Histórico

Por fim, o comando de histórico deveria ser implementado como forma de exibir os 50 últimos comandos executados.

1.2 Dificuldades encontradas

Durante a implementação da primeira parte do trabalho prático foram encontradas as seguintes dificuldades.

1.2.1 Permissões de usuário

O comportamento padrão do *shell* padrão do Linux é manter as permissões de arquivo quando a saída deste é redirecionada como criação de um novo arquivo:

```
matheus@note:~/Documentos/ufmg/so/sistemas-operacionais/trabalho-pratico-1$ cat teste.txt > teste1.txt
matheus@note:~/Documentos/ufmg/so/sistemas-operacionais/trabalho-pratico-1$ ls -la
total 292
drwxrwxr-x 3 matheus matheus 4096 set  8 19:28 .
drwxrwxr-x 4 matheus matheus 4096 ago 25 20:12 ..
drwxrwxr-x 3 matheus matheus 4096 set  8 19:07 basic-shell
-rwxrwxr-x 1 matheus matheus 17736 set  8 18:43 neutop
-rw-rw-r-- 1 matheus matheus 4477 set  8 18:44 neutop.c
-rwxrwxr-x 1 matheus matheus 16872 set  8 16:34 signaltester
-rw-rw-r-- 1 matheus matheus 427 mar 11 16:43 signaltester.c
-rw-rw-r-- 1 matheus matheus 6 set  8 19:28 teste1.txt
-rw-rw-r-- 1 matheus matheus 6 set  8 19:27 teste.txt
```

No exemplo da imagem, ao redirecionar o comando *cat* do arquivo *teste.txt* para o arquivo *teste1.txt*, as permissões de usuário foram preservadas.

Durante a implementação do *shell* esse comportamento não pôde ser reproduzido da forma correta e foi contornado para permitir o acesso ao arquivo com as flags: IRWXU, IRWXG, IRWXO que representam a capacidade de leitura escrita e execução para usuário, grupo e outros usuários respectivamente.

1.2.2 Histórico de comandos

Por algum motivo que exaustivamente foi testado, porém não foi solucionado, a lista de comandos sempre exibe os *n* comandos executados como o último comando executado.

Em outras palavras, caso o usuário execute a seguinte ordem: **ls**, **ls -la**, **ls -lrt**, **ps -edag** "pipe" **grep signaltest** e por fim execute o comando para exibir o histórico, serão exibidos 5 vezes o comando histórico.

```
matheus@note:~/Documentos/ufmg/so/sistemas-operacionais/trabalho-pratico-1/basic-shell$ ./sh
$ ls
grade.sh sh sh.c tests
$ ls
grade.sh sh sh.c tests
$ ls
grade.sh sh sh.c tests
$ ls
grade.sh sh sh.c tests
$ history
1 history
2 history
3 history
4 history
$
```

Apesar das instruções orientarem a limitar o histórico a 50 comandos, a implementação entregue limita a lista aos 5 últimos comandos definidos em código uma vez que a operação histórico não funciona corretamente.

1.3 Execução de testes

A execução dos testes disponibilizada falhou apenas em 1 dos 9 testes, sendo este o teste de presença de *warnings* durante a compilação.

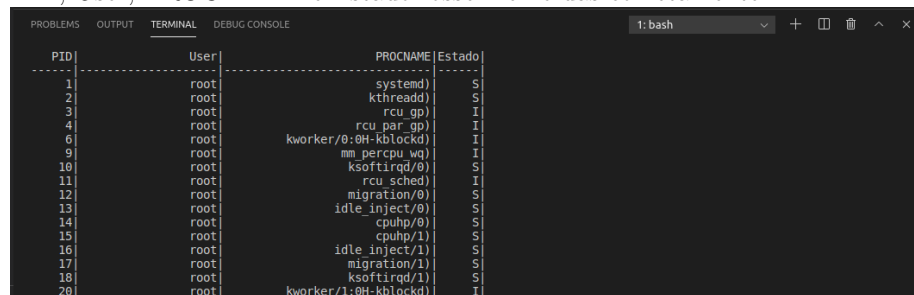
```
matheus@note:~/Documentos/ufmg/so/sistemas-operacionais/trabalho-pratico-1/basic-shell$ ./grade.sh
[0] compilation has warnings
your code passes 8 of 9 tests
check the output of the getmarks.py script on your code
```

2 TOP

A implementação do TOP seguiu as diretrizes para a utilização do *file handler* `/proc/` e seus subdiretórios.

A abordagem adotada utiliza um laço de repetição infinita assim como o exemplo no *shell* básico para ler os subdiretórios presentes e extrair as informações referentes aos processos em execução. Sua interrupção pode ser feita através de um `SIGHUP` (1) passando o PID do `meutop` como parâmetro, ou encerrando a execução com **CNTRL + C**.

Foram definidas funções para formatação do cabeçalho bem como das linhas de informação de cada processo para que a cada iteração do laço as informações PID, User, PROCNAME e Estado fossem exibidas corretamente.



PID	User	PROCNAME	Estado
1	root	systemd	S
2	root	kthreadd	S
3	root	rcu_gp	I
4	root	rcu_par_gp	I
6	root	kworker/0:0H-kblockd	I
9	root	mm_percpu_wq	I
10	root	ksoftirqd/0	S
11	root	rcu_sched	I
12	root	migration/0	S
13	root	idle_inject/0	S
14	root	cpuhp/0	S
15	root	cpuhp/1	S
16	root	idle_inject/1	S
17	root	migration/1	S
18	root	ksoftirqd/1	S
20	root	kworker/1:0H-kblockd	I

Para garantir a atualização das informações a cada segundo, a biblioteca `time.h` foi utilizada.

Para possibilitar a entrada de dados durante a execução do `top`, foi utilizada a biblioteca `sys/select.h` que permite monitorar em intervalos de tempo definidos os `textitfile` descriptors.

Assim como instruído, ao apresentar um sinal **SIGHUP** e o ID de um processo, a chamada de sistema *kill* é executada e o processo é terminado.

Exemplo de encerramento do TOP utilizando sinais

80051	root	kworker/u8:2-events unbound)	I
80146	root	kworker/1:0-events)	I
80168	root	kworker/3:2-events)	I
80264	matheus	chrome)	S
80276	matheus	chrome)	S
80420	root	kworker/u8:3-events unbound)	I
80461	root	kworker/1:1-events)	I
80462	root	kworker/3:0-mm_percpu_wq)	I
80561	root	kworker/2:1-events)	I
80564	root	kworker/u8:0)	I
80625	root	kworker/0:2-events)	I
80695	matheus	meutop)	R

80695 1
Desconexão
matheus@note:~/Documentos/ufmg/so/sistemas-operacionais/trabalho-pratico-1\$