Arquitectura e Sistemas de Computadores II

LEI - FCT/UNL - 2006/2007

Trabalho prático 7

Shell (4ª parte)

Objectivos Este trabalho visa continuar o desenvolvimento de um interpretador de comandos (*shell*) para UNIX, com vista ao exercício do controlo dos sinais que podem ser produzidos pelo utilizador ao terminal.

Bibliografia: "UNIX System Programming", cap. 6 e 7 do livro; manuais *on-line*.

Introdução

Pretende-se agora que o *shell* não termine a quando da combinação de teclas "Control+C" mas, em vez disso, permita terminar apenas o comando em execução no terminal. De forma idêntica, pretende-se que a combinação "Control+Z" possa ser usada para passar o comando em *foreground* para uma execução em *background*.

Se lhe simplificar o código, admita que este tratamento é feito apenas para a situação da execução de comandos sem redirecções ou *pipes*.

Sinais

Todos os processos criados a partir do seu *shell* são considerados como fazendo parte de um grupo de processos (*process group*). Os terminais podem enviar alguns sinais a todos os processos do grupo que usa o terminal por combinações de teclas pré-definidas. Normalmente, a combinação Ctrl+C envia o sinal SIGINT e a combinação Ctrl+Z, o sinal SIGTSTP. O comportamento por omissão de um processo perante estes sinais é, respectivamente, terminar e bloquear o processo. Tal significa que este vai ser o comportamento dos processos em execução no terminal, assim como do nosso próprio *shell*! Ora não é esse o comportamento esperado pelo utilizador. Quando se prime Ctrl+C pretende-se terminar o processo que executa no terminal mas não o *shell*. Quando se prime Ctrl+Z predende-se parar o processo no terminal, mas não o próprio *shell*. Note ainda que neste trabalho, para o caso do Ctrl+Z, pretende-se que o processo no terminal passe a executar em *background*.

Tratamento dos sinais

Deve começar por tornar o seu *shell* imune a estes sinais, pela declaração como sinais a ignorar (SIG_IGN) via a chamada ao sistema `signal`. Mas deve, para a execução dos restantes comandos, manter o comportamento por omissão (SIG_DFL), o que significa repor o comportamento normal antes de os executar via `execve` ou equivalente.

Na situação de um processo que executa em *foreground*, quando é premida a combinação Ctrl+Z, queremos que o *shell* detecte a situação, no seu `waitpid`, para parar de esperar pelo processo passando-o a *background*. Tal pode ser conseguido porque, indicando a *flag*

WUNTRACED, a chamada `waitpid` termina quando o processo por que esperamos for bloqueado pelo SIGTSTP. De seguida o *shell* deve continuar a sua execução, deixando agora o processo em *background*, ou seja, não aguarda pela sua terminação com `waitpid`.

Note no entanto que quer este processo, quer todos os outros que já estejam em *background*, receberam SIGTSTP e ficaram bloqueados. O seu *shell* deve enviar-lhes o sinal SIGCONT, para que os processos continuem a sua execução. No caso do processo em *foreground* tal pode ser detectado testando o *status* devolvido pelo `waitpid`. Exemplo:

```
if ( waitpid(pid, &status, WUNTRACED)<0 ) perror("waitpid");
else if ( WIFSTOPPED(status) ) kill( pid, SIGCONT );
```

Para tratar todos os restantes processos deve notar que, de cada vez que um processo termina ou é bloqueado pelo SIGTSTP, o processo pai é informado dessa situação via o sinal SIGCHLD. Este sinal é normalmente ignorado, mas poderemos declarar uma função de tratamento (*handler*) e mandar continuar todos os processos parados, para que estes não fiquem indefinidamente parados. Pode optar por obter todos os *status* e enviar SIGCONT aos processos bloqueados ou, numa só chamada ao sistema, pedir o envio do sinal SIGCONT a todos os processos do grupo. Tal pode ser conseguido usando 0 como identificador, exemplo:

```
kill( 0, SIGCONT );
```

Note que neste sistema, o *handler* é perdido a quando do sinal, devendo a função de tratamento voltar a associar a função que se pretende ao sinal, via `signal`.

Opcional:

Quando prime Ctrl+C todos os processos recebem o sinal e terminam, quer estejam em *foreground* ou em *background*. Normalmente o comportamento pretendido é que apenas termine o processo em *foreground* no terminal e, os restantes em *background*, continuem a sua execução. Resolva este problema por forma a que os processos lançados em *background* não terminem quando for feito Ctrl+C. A mesma técnica pode ser aplicada para impedir os processos em *background* de bloquearem com o sinal SIGTSTP. Note: Pode tentar uma solução apenas usando os conhecimentos até agora adquiridos, mas a verdadeira solução para este tipo de problema passa pela utilização de vários grupos de processos e pela afectação ao terminal apenas do grupo em *foreground*. Pode ver os manuais de `tcsetpgrp` e `setpgrp`.