

SISTEMAS OPERATIVOS I

Controlo de processos em UNIX

Revisto em Setembro de 2004

Sandra Machado

Berta Batista

Sugestões e participações de erros para: bbatista@dei.isep.ipp.pt

CONTROLO DE PROCESSOS EM UNIX

Índice

1	Controlo de Processos em UNIX	3
1.1	Background vs Foreground.....	3
1.2	Executar Processos em Background; PIDs.....	3
1.3	Standard Input, Standard Output e Standard Error	3
1.4	O comando Wait.....	4
1.5	O comando PS - Process Status.....	4
1.6	O comando Kill.....	4
1.7	Prioridade de um Processo.....	5
1.8	Tornar um Processo Imune ao Logout	5
1.9	Executar Comandos a Determinado Dia em Determinada Hora.....	5
1.9.1	Executar Comandos Quando o Sistema Está Pouco Ocupado - batch	5
1.9.2	Executar Comandos Numa Altura Especificada - at	6
1.9.3	Executar Comandos Repetidamente a uma Hora Específica - crontab.....	7
1.10	QUESTÕES1 (para resolver na aula)	7
1.11	QUESTÕES2 (para resolver em casa)	8

1 Controlo de Processos em UNIX

1.1 *Background vs Foreground*

Com o processamento em *foreground* a shell (*parent process*) espera que o comando (*child process*) seja executado antes de exibir uma nova *prompt*.

Pelo contrário o processamento em *Background* permite que sejam executados vários processos simultaneamente e que novos comandos sejam iniciados sem que os anteriores tenham terminado a sua execução.

Apesar da multi-tarefa ter consistido numa das mais importantes características do Unix, a vulgarização dos terminais gráficos tornou-a menos essencial (porque quando o utilizador quer iniciar um processo antes de terminar o anterior, só precisa de abrir outra janela e escrever a nova linha de comandos).

Há, no entanto, situações em que existem vantagens no processamento em *background*.

1.2 *Executar Processos em Background; PIDs*

Basta colocar o caracter **&** no fim da linha de comandos.

```
$ find /bin /etc -name passwd 2>/dev/null &
12283
$
```

(12283 é o PID - Process ID Number do processo lançado em background)

O Process ID Number é a resposta da shell à linha de comandos terminada em **&**. . Pode ser visto como o bilhete de identidade do processo a correr em *background*.

1.3 *Standard Input, Standard Output e Standard Error*

Executar comandos em *background* não redirecciona automaticamente o *standard input*, *standard output* e *standard error*, embora seja aconselhável que o utilizador o faça.

Se o *std input* não for redireccionado, tanto a shell como o comando vão estar à espera de ler o teclado e não há possibilidade de se saber que caracteres são *input* do comando e que caracteres são uma nova linha de comandos.

Se o *std output* o *std error* não forem redireccionados para um ficheiro, o utilizador será interrompido pelo resultado da execução dos comandos e pelo possível erro, enquanto pretende escrever outras linhas de comandos, o que causará uma enorme confusão. Assim aconselha-se que para lançar processos em *background* se recorra a:

```
$ (linha de comandos < ficheiro_in > ficheiro_out) >2 /dev/null &
```

1.4 O comando Wait

O comando `wait` obriga a shell a esperar a execução de todos os comandos em background, antes de exibir uma nova prompt. Assim se se invocar este comando a prompt da shell só será apresentada depois de todos os comandos terminarem.

```
$ wait
```

1.5 O comando PS - Process Status

O comando `ps` permite ao utilizador saber o estado de um processo. A sua sintaxe genérica é:

```
$ ps [-lef] [-t lista_terminais][--u lista_utilizadores][--p lista_processos]
```

Opções:

- l – long list
- f – full list
- e – environment

Na sua forma mais simples, isto é sem argumentos, permite ao utilizador ver o estado dos seus processos (incluindo os que estão em background).

```
$ alvega> ps
  PID TTY          TIME CMD
 23455 pts/0    00:00:00 csh
 23472 pts/0    00:00:00 ps
alvega>
```

```
alvega> ps -l
  F S  UID     PID  PPID  C  PRI  NI ADDR      SZ  WCHAN  TTY          TIME CMD
 100 S  1017  23455 23454  0   71   0  -   1089  rt_sig pts/0    00:00:00 csh
 000 R  1017  23479 23455  0   78   0  -   1489  -      pts/0    00:00:00 ps
alvega>
```

1.6 O comando Kill

O comando `kill` é usado para enviar sinais para o processo em background.

Um sinal é uma mensagem simples que avisa o processo da ocorrência de algum evento anormal. Os sinais podem causar:

- terminar o processo;
- suspender o processo;
- não fazer nada.

A sua sintaxe genérica é:

```
$ Kill [-l] [sinal] pid
```

A opção `-l` apenas funciona como "help" e devolve a lista de todos os sinais do sistema. Os sinais apresentados abaixo, têm por função terminar o processo. Se não for especificado nenhum sinal é assumido o 15 ou TERM.

- 1 – HUP – Hangup
- 9 – KILL – termina o processo
- 15- TERM – Default

```
alvega> kill -l
HUP INT QUIT ILL TRAP ABRT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM
STKFLT
CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH POLL PWR SYS
RTMIN RTMIN+1 RTMIN+2 RTMIN+3 RTMAX-3 RTMAX-2 RTMAX-1 RTMAX
alvega>
```

1.7 Prioridade de um Processo

A prioridade de um processo é calculada pelo sistema operativo e determina que processo deve ser executado a seguir. O cálculo é feito com base na utilização do CPU e com o nice value.

A prioridade do processo não pode ser alterada, mas podemos afectar o seu nice value com o comando **nice**. O nice value é um número de 0 a 20. O 0 é só um bocadinho simpático e 20 é realmente simpático.

O comando nice é vulgarmente utilizado com processos a correr em background, mas também pode ser usado para processos em foreground.

```
$ nice [-n°] linha de comandos [&]
```

1.8 Tornar um Processo Imune ao Logout

Colocar um comando em background não o torna imune ao logout. Esta característica pode consistir numa limitação grave.

Existe um comando de Unix que executa os comandos especificados como parâmetros, de modo a que ignorem logouts. É o comando **nohup**. O nohup pode ser usado com processos em background e em foreground, embora seja mais utilizado para os primeiros.

```
$ nohup linha de comandos [&]
```

1.9 Executar Comandos a Determinado Dia em Determinada Hora

Além de permitir executar comandos em background, o Unix permite executar comandos em determinado dia a determinada hora especificada pelo utilizador / administrador.

1.9.1 Executar Comandos Quando o Sistema Está Pouco Ocupado - batch

```
$ batch
comando1
comando2
...
comandon
^D
```

O comando batch significa: "Quando não estiveres tão ocupado, executa-me o comando1, comando2, ..., comando n, por favor".

O comando batch lê o standard input (sequência de comandos, um por linha, terminada com ^D), mas também pode ser o fim de uma pipeline (...|...|batch) ou pode ter os comandos redireccionados de um ficheiro (batch < ficheiro_comandos).

Todos os comandos têm um jobnumber, exibido quando a shell lê a linha de comandos.

Quando a performance do sistema atinge determinado nível predefinido, os comandos são executados tal como se tivessem sido executados pelo utilizador interactivamente, mantendo-se assim características como a shell, o *environment* (ambiente de trabalho) e a umask.

O output é enviado por mail ao utilizador ou é guardado num ficheiro especificado, caso o standard output seja redireccionado. O standard error tem tratamento idêntico.

1.9.2 Executar Comandos Numa Altura Especificada - at

```
at hora [data][+incremento]
comando1
comando2
...
comandon
^D
```

Como o batch, o comando at também pode ler o standard input (sequência de comandos, um por linha, terminada com ^D), ser o final de uma pipeline (...|...| at 8:50am), ou ter o standard input redireccionado por um ficheiro (at 4:30pm).

Todos os comandos têm um jobnumber, exibido quando a shell lê a linha de comandos.

Os argumentos do comando podem ser apenas a hora a que o comando deve ser executado, no dia corrente, ou uma sequência de hora e data ou um incremento a determinada data especificada.

Argumentos válidos são:

at 9	Executado às 9 horas da manhã do dia corrente
at 13	Executado à 1 da tarde do dia corrente
at 0643	Executado às 6:43 horas da manhã do dia corrente
at 0643pm	Executado às 6:43 horas da tarde do dia corrente
at 6:43pm	Executado às 6:43 horas da tarde do dia corrente
at noon	Executado às 12 horas do dia corrente
at 1zulu	Executado à 1 hora da manhã do dia corrente no sistema horário GMT
at 07:05am Tuesday	Executado às 7:05 horas da manhã na próxima Terça-feira
at 07:05am March 12	Executado às 7:05 horas da manhã do dia 12 de Março do ano corrente
at 13 Tomorrow	Executado à 1 da tarde de amanhã
at January 1 2002	Executado à 1 da manhã do dia 1 de janeiro de 2002
at 1 January 1, 2002	Executado à 1 da manhã do dia 1 de janeiro de 2002
at noon + 1 day	Executado ao meio dia de amanhã
at now + 2 Weeks	Executado à hora corrente a duas semanas do dia corrente.

```
at -l [jobnumber]
```

```
at -r jobnumber
```

Opções:

- l - lista todos os trabalhos at e batch;
- r - remove determinado trabalho at ou batch.

Se o standard output e o standard error não forem redireccionados, serão enviados por mail, depois de executados aproximadamente à hora determinada. Os comandos são executados como se fossem executados pelo utilizador interactivamente.

Nem todos os utilizadores do sistema têm permissão para usar o at. A lista de pessoas com permissão é guardada no ficheiro mantido pelo administrador /usr/lib/cron/at.allow. Ou então existe o ficheiro, também mantido pelo administrador, /usr/lib/cron/at.deny, que especifica quem não tem acesso.

1.9.3 Executar Comandos Repetidamente a uma Hora Específica - crontab

```
$ crontab [ficheiro_crontab]
$ crontab -l
$ crontab -r
```

O comando crontab lê uma tabela fornecida por um ficheiro, pipeline ou pelo standard input.

A tabela tem um número de linhas correspondentes aos comandos que o utilizador pretende que sejam executados e cada linha tem obrigatoriamente 6 colunas:

- 1 - minutos - gama de 0 a 59;
- 2 - horas - gama de 0 a 23;
- 3 - dia do mês - gama de 1 a 31;
- 4 - mês do ano - gama de 1 a 12;
- 5 - dia da semana - gama de 0 a 6;
- 6 - comando Unix, shell script ou programa executável.

Exemplo:

```
30 7 * * 1-5 calendar
10 8 * * 1 write i000...%BOM DIA%BOA SEMANA
```

No exemplo usam-se os caracteres * e %. O primeiro significa todos os números possíveis da gama admissível, no caso significa todos os dias do mês e todos os meses do ano. O % é substituído por uma nova linha. Ou seja, quando o primeiro % surge significa que o comando está à esquerda e à direita está o standard input.

Todos os comandos da cron table são executados aproximadamente nas data e hora especificadas e se o standard output e error não forem redireccionados, serão devolvidos ao utilizador por mail. Mas são todos executados na /bin/sh.

As opções são idênticas às do at.

1.10 QUESTÕES1 (para resolver na aula)

1. Execute os comandos seguintes em *background*:
2. `find / -size +200000c -exec ls -l \{\} \;`
3. `find / -mtime -5 -exec file \{\} \;`
4. Use o comando **ps** para ver se os comandos do exercício anterior ainda estão a correr. Teste a opção "**f**".
5. Mate o primeiro dos dois processos que foram colocados a correr no exercício 9.
6. Espere que termine o segundo comando do exercício 9.
7. Execute o primeiro comando do exercício 9 novamente, mas com um prioridade mais baixa.
8. Execute o segundo comando do exercício 9 de modo a que fique em execução após terminar a sua sessão de trabalho.
9. Termine a sessão de trabalho e inicie novamente. Verifique se os comandos dos exercícios anteriores estão ou não a correr.

1.11 QUESTÕES2 (para resolver em casa)

10. Qual a diferença entre processos a correr em *background* e *foreground*.
11. Como se pode colocar um processo a correr em *background*.
12. que acontece ao "*standard input*" e ao "*standard output*" de um comando a correr em *background*.
13. Explique o uso do comando **wait**.
14. Qual o comando que lista os processos a correr? Explique as suas opções.
15. Explique a funcionalidade do comando **kill** assim como as suas opções.
16. Qual o comando UNIX que permite modificar a prioridade de um processo.
17. Como se pode criar um processo que continue a ser executado depois de terminar a sua sessão trabalho.