Sistemas Operativos
Gestão de Processos

Grupo de Sistemas Distribuídos
Universidade do Minho

1 Objectivos

Familiarizar-se e utilizar as chamadas ao sistema relativas a criação e gestão de processos.

2 Chamadas ao sistema

```
#include <unistd.h>     /* chamadas ao sistema: defs e decls essenciais */
#include <sys/wait.h>   /* chamadas wait*() e macros relacionadas */

pid_t getpid(void);
pid_t getppid(void);
pid_t fork(void);
void _exit(int status);
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
int WIFEXITED(int status); /* macro */
int WEXITSTATUS(int status); /* macro */
```

3 Exercícios propostos

1. Implemente um programa que imprima o seu identificador de processo e o do seu pai. Comprove – invocando o comando `ps` – que o pai do seu processo é o interpretador de comandos que utilizou para o executar.

2. Implemente um programa que crie um processo filho. Pai e filho devem imprimir o seu identificador de processo e o do seu pai. O pai deve ainda imprimir o pid do seu filho. Porventura poderá confrontar-se com a situação em que o pai do processo filho é apresentado como sendo o pid 1. Recorde que o pai pode terminar a sua execução antes do processo filho: nesse caso, o filho diz-se *órfão*, sendo adoptado pelo processo `init`, cujo pid é efectivamente 1. Num outro terminal – invocando o comando `ps l` – poderá ainda confrontar-se com a situação em que o processo filho é assinalado como estando num estado `Z`, designado por *zombie*. Este caso ocorre quando um processo termina e o seu pai não recolheu ainda a correspondente informação (através da invocação da chamada `wait()`).

3. Implemente um programa que crie dez processos filhos que deverão executar sequencialmente. Para este efeito, os filhos podem imprimir o seu pid e o do seu pai, e finalmente, terminarem a sua execução com um valor de saída igual ao seu número de ordem (e.g.: primeiro filho criado termina com o valor `1`). O

pai deverá imprimir o código de saída de cada um dos seus filhos. Note que só pode imprimir o código de saída de um filho se este tiver terminado com a invocação (implícita ou explícita) da chamada ao sistema `_exit()`.

4. Implemente um programa que crie dez processos filhos que deverão executar (potencialmente) em concorrência. O pai deverá esperar pelo fim da execução de todos os seus filhos, imprimindo os respectivos códigos de saída.

5. Implemente um programa que crie uma descendência em profundidade de dez processos, ou seja, o processo cria um filho, este filho cria outro, e assim por diante até ao décimo nível de descendência. Cada processo deverá imprimir o seu pid e o pid do seu pai. Se desejar, poderá obrigar cada processo a esperar pelo fim da execução do seu (eventualmente) único filho.

6. Pretende-se determinar a existência de um determinado número inteiro nas linhas de numa matriz de números inteiros, em que o número de colunas é muito maior do que o número de linhas. Implemente, utilizando processos um programa que determine a existência de um determinado número, recebido como argumento, numa matriz gerada aleatoriamente.

7. A partir do cenário descrito no exercício anterior, pretende-se que imprima por ordem crescente os números de linha onde existem ocorrências do número.

para devolver ao pai através de variavel, usar no filho o _exit com o valor e o pai usa o wexitstatus