Sistemas Operativos
Execução de Programas

Grupo de Sistemas Distribuídos
Universidade do Minho

1 Objectivos

Familiarizar-se e utilizar as chamadas ao sistema relativas à execução de programas.

2 Chamadas ao sistema

```
#include <unistd.h>     /* chamadas ao sistema: defs e decls essenciais */

int execl(const char *path, const char *arg0, ..., NULL);
int execlp(const char *file, const char *arg0, ..., NULL);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
```

3 Exercícios propostos

1. Implemente um programa que execute o comando `ls -l`. Note que no caso da execução ser bem sucedida, mais nenhuma outra instrução é executada do programa original.
2. Implemente um programa semelhante ao anterior que execute o mesmo comando mas agora no contexto de um processo filho.
3. Implemente um programa que imprima a lista de argumentos recebidos na sua linha de comando.
4. Implemente um programa que execute o programa da questão anterior com uma qualquer lista de argumentos. Mantendo o nome do ficheiro que corresponde o programa executável, experimente alterar o primeiro elemento da lista de argumentos (índice zero do `argv`).
5. Implemente um programa que execute concorrentemente uma lista de executáveis especificados como argumentos da linha de comando. Considere os executáveis sem quaisquer argumentos próprios. O programa deverá esperar pelo fim da execução de todos processos por si criados.
6. Implemente uma versão simplificada da função `system()`. Ao contrário da função original, não tente suportar qualquer tipo de redireccionamento, ou composição/encadeamento de programas executáveis. O único argumento deverá ser uma *string* que especifica um programa executável e uma eventual lista de argumentos. Procure que o comportamento e valor de retorno da sua função sejam compatíveis com a original.

7. Implemente um interpretador de comandos muito simples ainda que inspirado na `bash`. O interpretador deverá executar comandos especificados numa linha de texto introduzida pelo utilizador. Os comandos são compostos pelo nome do programa a executar e uma eventual lista de argumentos. Os comandos podem ainda executar em primeiro plano, ou em pano de fundo, caso o utilizador termine a linha com `&`. O interpretador deverá terminar a sua execução quando o utilizador invocar o comando interno `exit` ou quando assinalar o fim de ficheiro (`Control-D` no início de uma linha em sistemas baseados em Unix).

4 Exercícios Adicionais

1. Implemente um programa `controlador` que execute concorrentemente um conjunto de programas especificados como argumento da sua linha de comando. O `controlador` deverá re-executar cada programa enquanto não terminar com código de saída nulo. No final da sua execução, o `controlador` deverá imprimir o número de vezes que cada programa foi executado. Considere que os programas são especificados sem qualquer argumento.

```
$ ./controlador a.out b.out c.out
a.out 2
b.out 4
c.out 4
```

Sugestão: Para testar o `controlador` implemente um programa auxiliar que termine imediatamente com um código de saída aleatório entre 0 e 3. Por exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char *argv[]) {
    int res;
    srandom(time(NULL));
    res = random() % 3;
    printf("%s: %d\n", argv[0], res);
    return res;
}
```

E compile:

```
$ gcc -Wall auxiliar.c -o a.out
$ gcc -Wall auxiliar.c -o b.out
$ gcc -Wall auxiliar.c -o c.out
```