1

Sistemas Operativos

Trabalho prático 2

Versão 2.1.0

1. Introdução

O presente trabalho prático visa uma maior familiarização com a programação de mecanismos utilizados em sistemas operativos. É realizado em grupo. Esses grupos já estão definidos na unidade curricular. O trabalho está sujeito a apresentação e defesa, realizada individualmente por cada aluno. As defesas serão marcadas em data oportuna, comunicadas aos alunos e publicadas na plataforma de e-learning.

É, obviamente, interdita a cópia parcial ou integral de trabalhos e, a ser detetada, conduzirá à adequada penalização dos envolvidos. O trabalho deverá apresentar-se na forma de código fonte e de um relatório claro e conciso, que também será objeto de avaliação.

A aplicação deverá ser desenvolvida em linguagem C, no ambiente de desenvolvimento utilizado nas aulas práticas, tendo em atenção as boas práticas de programação.

2. Descrição do problema

Desenvolva uma aplicação multiprocesso que escreva o código de um programa que permita criar a hierarquia de processos da figura seguinte, de acordo com o definido na secção 3.

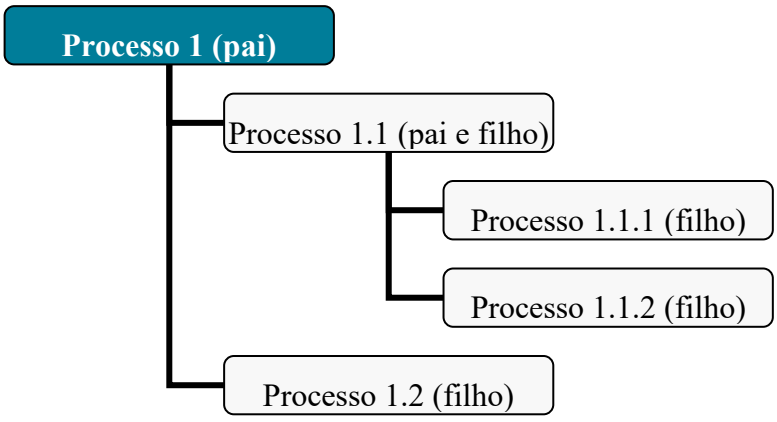


Figura 1 – Hierarquia de processos

3. Implementação

A aplicação deve apresentar um menu do tipo mostrado abaixo e obedecer aos seguintes requisitos funcionais:

1. Criar ficheiros
2. Mostrar valores
3. Eliminar ficheiros
4. Terminar

3.1 Criar ficheiros

3.1.1 A opção 1 deve criar a hierarquia de processos representada na Figura 1

3.1.2 Cada processo deve criar um ficheiro com o nome PID.pso, em que PID representa o pid do próprio processo.

3.1.3 Escrever no ficheiro o seu PID e o PID do seu pai, utilizando a notação [PID Pai, PID `Filho`], de modo a permitir a verificação visual da hierarquia de processos criada.

3.1.4 Escrever o nome do grupo, tipo PxGy, em que x representa o nº do turno e y o nº do grupo.

3.1.5 Escrever um texto com comprimento aleatório.

3.2 Mostrar valores

3.2.1 A opção 2 deve permitir percorrer a diretoria para obter o nome dos ficheiros e apresentar no stdout os dados de cada ficheiro, da seguinte forma:

Ficheiro: nome XX nºdebytes

3.2.2 Devem ser criados tantos processos quanto o número de ficheiros existente e, em paralelo, cada um dos processos faz a contagem do número de bytes de cada ficheiro.

3.3 Eliminar ficheiros

A opção 3 deve permitir eliminar os ficheiros criados na opção 1.

3.4 Terminar

A opção 4 deve terminar o programa, criando um processo que envie ao processo pai o sinal *SIGINT*. O processo pai deve fazer o devido tratamento do sinal *SIGINT*, de forma a terminar usando o sinal *SIGKILL*.

3.5 Funções

Para o desenvolvimento do programa, deve utilizar funções tais como *open, read, write, getpid, getppid, fork, wait, kill, signal ou sigaction*, entre outras.

Para ler o conteúdo da diretoria, pode utilizar a função *scandir* ou *readdir* ou outra que julgue adequada. A utilização de *scandir* mostra-se no exemplo seguinte:

```
// Definição da estrutura que guarda os nomes dos ficheiros de uma pasta
```

```
struct dirent **namelist;
//Função para ler a diretoria
  n = scandir(".", &namelist,filtro, alphasort);
```

4. Relatório

O relatório, que se pretende breve, deverá justificar as opções tomadas, bem como eventuais desvios relativamente às especificações constantes deste enunciado. Devem ser identificadas as principais dificuldades encontradas e respetivas soluções (quando não mencionadas neste enunciado). No caso de o trabalho entregue não implementar todos as especificações referidas, as respetivas lacunas deverão necessariamente fazer parte desse relatório.

5. Entrega dos Trabalhos

O programa deve ser enviado para a plataforma de e-learning (https://moodle.estgv.ipv.pt), assim como um relatório com a descrição do programa implementado, respetivas limitações e problemas.

A data limite de entrega está definida nesta atividade de submissão do trabalho. Os trabalhos entregues depois dessa data, não são considerados.

6. Updates deste enunciado

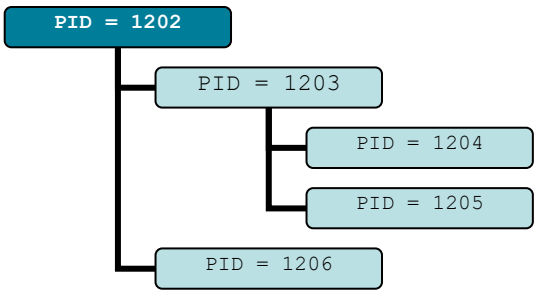
De forma a viabilizar a eventual introdução de atualizações a este documento, chama-se a atenção de que é requisito deste trabalho a verificação da existência de versões atualizadas do enunciado. Nesse sentido, ao enunciado está associada uma versão.

Esta é a versão v2.1.0.

7. Cenários de utilização - Simulação

De seguida apresenta-se um exemplo de execução possível para o programa (os PIDs aqui apresentados, servem, apenas, de exemplo).

```
$ ./myfork
[Pai, Filho] = [1203, 1204]
[Pai, Filho] = [1203, 1205]
[Pai, Filho] = [1202, 1206]
[Pai, Filho] = [1202, 1203]
[Pai, Filho] = [7201, 1202]
```



Para realizar as tarefas pretendidas deverá utilizar as funções estudadas no âmbito da programação de processos, tais como fork, getpid, getppid, wait, …

1. **Menu**

```
quental@viriato:~/tp02$ tp2

1. Criar ficheiros
2. Mostrar valores
3. Eliminar ficheiros
4. Terminar
```

2. **Escolher opção 2 – Mostrar valores**

```
1. Criar ficheiros
2. Mostrar valores
3. Eliminar ficheiros
4. Terminar
2
 Não existem ficheiros .pso na pasta
```

3. **Escolher opção 1 – Criar ficheiros**

   De notar que os PIDs apresentados servem, apenas, para visualizar os resultados. Os dados são guardados nos ficheiros respetivos.

```
1. Criar ficheiros
2. Mostrar valores
3. Eliminar ficheiros
4. Terminar
1
[Pai, Filho] = [279435,279437]
[Pai, Filho] = [279420,279435]
[Pai, Filho] = [279435,279436]
[Pai, Filho] = [279420,279438]
[Pai, Filho] = [188985,279420]
```

Conteúdo da pasta:

```
quental@viriato:~/tp02$ ls
279420.pso  279435.pso  279436.pso  279437.pso  279438.pso  comum.h  tp2  tp2.c
quental@viriato:~/tp02$
```

Conteúdo do ficheiro 279420.pso, por exemplo:

```
quental@viriato:~/tp02$ cat 279420.pso
[PAI 188985 FILHO 279420]
P1G1
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the in
dustry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and s
crambled it to make a type quental@viriato:~/tp02$
```

4. **Escolher opção 2 – Mostrar valores**

```
1. Criar ficheiros
2. Mostrar valores
3. Eliminar ficheiros
4. Terminar
2
Ficheiro: 279438.pso 130 bytes
Ficheiro: 279437.pso 229 bytes
Ficheiro: 279436.pso 81 bytes
Ficheiro: 279435.pso 159 bytes
Ficheiro: 279420.pso 263 bytes
```

5. **Escolher opção 4 - Terminar**

```
1. Criar ficheiros
2. Mostrar valores
3. Eliminar ficheiros
4. Terminar
4
Killed
```

Bom trabalho!!