

Sistemas Operativos 2022/2023

Ficha Nº 3

Criação e gestão de processos. Redireccionamento

Tópicos ->

Gestão de processos em linha de comandos.
Criação de processos com *fork*. Execução de programas com *exec*.
Redireccionamento

Exercícios

Exercícios 1 a 3: Focam os conceitos de execução em *background* e *foreground*, jobs e os comandos associados.

Exercícios 4 e 6: Exemplificam o uso de *fork* e *exec*.

Exercícios 7: Foca o uso de redireccionamento.



1. Elabore um programa que peça uma cadeia de caracteres ao utilizador e depois a imprima no ecrã. O pedido da cadeia de caracteres ao utilizador é apenas uma forma rápida de garantir que o programa não termina logo e fica à espera do utilizador.
 - a) Use dois terminais. No primeiro lance a execução do programa. Não responda ao pedido de caracteres e deixe o programa à espera. No segundo terminal verifique quais os programas que estão a correr.
 - I. Use o comando **ps** para ver os processos em execução.
 - II. Use o comando **ps** com a opção **a** e depois com a opção **ax**.
 - III. Veja que outras opções o comando **ps** tem através das páginas de manual (comando **man**).
 - b) No primeiro terminal, forneça os caracteres que programa está a pedir de forma a que este termine.

Objectivos do exercício - Processos em execução, comando **ps**

- Compreender como pode obter informação de processos em execução.
- Comandos **ps**, **man**.



2. Este exercício usa o programa do exercício anterior, mas no contexto de apenas um terminal.
 - a) Lance o programa em *background*. Use o caracter **&** na linha de comandos para este efeito: o programa foi lançado mas voltou imediatamente a ter acesso à linha de comandos e os caracteres que introduzir já não serão enviados ao programa (que ainda está em execução).

- b) Verifique que programas estão a correr (comando **ps**). Confirme que o seu programa está de facto em execução. Precisa de especificar a opção **a** no comando **ps**? Porque não?
- c) Usando o comando **jobs** verifique quais as tarefas que estão em execução.
- d) Traga a execução do seu programa novamente para *foreground* através do comando **fg**.
- e) Suspenda a execução do processo com a combinação de teclas **^Z**.
- f) Verifique com o comando **ps** que o processo ainda existe.
- g) Traga o processo para execução em background com o comando **bg**. Consegue fornecer os caracteres que o programa está a pedir?
- h) Traga o processo para execução **foreground** e forneça os caracteres que ele precisa para o encerrar.

Objectivos do exercício - Processos em execução. Gestão de jobs

- Compreender que os processos podem estar a correr em *foreground*, em *background*, e suspensos.
- Compreender as diferenças entre execução em *foreground* e execução em *background*.
- Compreender o conceito de *jobs* e sua gestão.
- Comandos **ps**, **jobs**, **fg**, **fb**.



3. Elabore um programa que imprime um número **N** de asteriscos com uma pausa de **S** segundos entre cada asterisco. O número **N** e **S** são passados na linha de comandos. O programa deve verificar se os números **N** e **S** foram efectivamente indicados na linha de comandos.

Neste exercício é importante que o output dos *printf* apareça imediatamente (normalmente só aparece quando o seu *buffer* interno enche ou quando aparece um “\n”). Para forçar o output a aparecer imediatamente invoque **fflush(stdout)**; após cada *printf*, ou melhor ainda, configure um *buffer* nulo com a chamada **setbuf(stdout, NULL)**; (sendo necessário fazer essa invocação apenas uma vez).

Vários exercícios nesta ficha e nas seguintes partilham este requisito. **Anote esta informação no seu caderno para não se esquecer.**

- a) Execute e teste o programa com 5 asteriscos e pausa de 1 segundo.
- b) Execute o programa em background com 20 asteriscos e uma pausa de 1 segundo. Deixe o programa a executar e passe para as alíneas seguintes.
- c) Confirme que os asteriscos continuam a executar. Averigüe quais os processos e quais as tarefas (*jobs*) em execução.
- d) Traga o programa para execução em *foreground*. Note que a execução do programa continua inalterada.
- e) (Se o programa já tiver terminado, lance novamente a sua execução.) Suspenda a execução do programa com **^Z**. Note que os asteriscos deixaram de aparecer.
- f) Verifique que o processo continua a existir. Verifique quais as tarefas (*jobs*) existentes.
- g) Traga o programa para execução em *background*. Note que os asteriscos recomeçaram.
- h) Traga a execução do programa para *foreground* e deixe-o terminar normalmente.
- i) Lance novamente o programa, mas agora em *background*. Use o carácter **&** na linha de comandos para este efeito. Verifique que o programa foi lançado e executa normalmente, imprimindo asteriscos, mas o utilizador tem acesso à linha de comandos. Experimente trazer o programa para execução *foreground*, depois suspendê-lo, e depois para execução em background novamente.

- j) Repita este exercício e o anterior até entender completamente: i) as diferenças entre execução em *foreground* e execução em *background*, II) comandos *jobs*, *fg*, *bg*, *ps*.

Objectivos do exercício - Processos em execução. Gestão de tarefas (jobs)

- Consolidar conceitos de execução em *background*, *foreground* e comandos associados.
- Comandos *ps*, *jobs*, *fg*, *fb*.



4. Construa três programas, de acordo com a especificação abaixo.

- Programa **port**. Este programa escreve “ola mundo” e termina.
- Programa **ing**. Este programa escreve “hello world” e termina.
- Programa **executa**. Este programa faz: i) Pede uma cadeia de caracteres ao utilizador. ii) se a cadeia de caracteres for “port”, executa o programa **port**. Se for “ingl”, executa o programa **ingl**, iii) escreve “missão cumprida” e termina. Utilize as funções **execl** ou **execlp**.

Nota: Este exercício também exige que o output dos *printf* apareça imediatamente. Utilize as recomendações dadas no exercício anterior. Se o output dos *printf* não aparecer imediatamente, os programas dariam a ideia que se estavam a executar por uma ordem diferente da real. Ouça a explicação do professor sobre este assunto.

- Lance o programa **executa** e experimente-o com “port” e com “ing”. Explique porque é que ao aparecer a mensagem de “ola mundo” ou “hello world”, já não aparece a mensagem “missão cumprida”. Explique o que acontece ao programa **executa** quando executa **port** ou **ing**.
- Experimente usar ambas as funções **execl** e **execlp** no programa **executa**. Em cada um dos casos teste-o indicando “port” e “./port”. Algumas das combinações **execl/execlp** e “port”/ “./port” levam a que o programa **port** não seja executado. Porquê? Anote as suas conclusões.
- Mude o código do programa de forma a passar directamente às funções **execl/execlp** os caracteres que foram escritos pelo utilizador. Experimente novamente:
 - Com “port”.
 - Com “./port”.
 - Com “ls”.

Explique o resultado para cada um dos anteriores com **execl** e **execlp**.

- Verifique com o programa **man** as outras funções da família **exec**. Repare nas diferenças entre elas e anote essa informação.

Objectivos do exercício - Execução de programas com funções da família exec.

- Compreender o que fazem as funções *exec*.
- Compreender como funciona a execução de programas em Unix.
- Compreender a diferença entre processo e programa.
- Entender as diferenças entre *execl* e *execlp*.
- Funções *execl* e *execlp* e restantes da família *exec*.



5. Construa o programa cujo função *main* tem o apenas o código apresentado abaixo. Caso precise, utilize o comando *man* para obter informações adicionais acerca do uso de *fork*.

```
int a = 10;
if (fork() == 0)
    a++;
else
    a--;
printf("\na = %d\n", a);
return 0;
```

- Tente prever o que aparecerá no ecrã antes de o executar.
- Execute o programa e explique o que aparece no ecrã.
- Verifique que ouviu e entendeu a explicação do professor acerca da função *fork*. Aponte essa informação.
- Proponha usos para a função *fork* (em que é que pode ser útil?).

Objectivos do exercício - Função *fork*

- Compreender o que faz a função *fork*.
- Compreender a utilidade da função *fork*.



6. Este exercício envolve 3 programas.

- Construa um programa *ding*. Este programa obtém uma palavra portuguesa como argumento pela linha de comandos e apresenta no ecrã a sua tradução em inglês (ou “unknown”) se não a souber traduzir. O seu programa irá saber traduzir 5 palavras; usará duas matrizes de cadeias de caracteres e um ciclo for para procurar a palavra, usando a função *strcmp*.

Duração desta alínea: 7 minutos. Se sentir muita dificuldade a fazer esta alínea deverá rever IP e Programação (ambas 1º ano) com muita urgência.

- Copie o ficheiro de código fonte para um novo ficheiro e, adaptando-o, construa o programa *dfran* que consegue traduzir palavras de português para francês (duração da alínea: 2 minutos).
- Escreva um programa *traduz*. Este programa tem um ciclo. Em cada iteração faz:
 - Pede uma letra e depois uma cadeia de caracteres ao utilizador.
 - Se a letra for “x”, termina (e já nem pede a cadeia de caracteres).
 - Se a letra for “i”, usa o programa *ding* para fazer com que apareça no ecrã a palavra traduzida em inglês.
 - Se a letra for “f”, usa o programa *dfran* para fazer com que apareça no ecrã a palavra traduzida em francês.

(Duração da alínea: 10 minutos).

Objectivos do exercício - Execução de programas com funções da família *exec*.

- Consolidar conhecimentos de fork, exec e de passagem de argumentos para programas.



7. Este exercício envolve é muito semelhante ao anterior. Copie o código dos programas desse exercício para novos ficheiros e adapte-os para obter novos programas muito parecidos com os anteriores.
- a) Oiça a explicação acerca de redireccionamento e anote no seu caderno o esquema que o professor colocou no quadro. Verifique que entende a diferença entre redireccionamento simples para ficheiros e redireccionamento entre dois programas com *pipes* anónimos.
 - b) Programa ***rding***. Este programa obtém uma palavra pede uma palavra ao utilizador (gets/scanf), e assumindo que é uma palavra portuguesa apresenta a sua tradução em inglês no ecrã.
 - c) Programa ***rdfran***. Tal como no caso anterior, a única diferença para o exercício anterior é que a palavra é obtida via gets/scanf e não como argumento de linha de comandos.
 - d) Programa ***rtraduz***. Cumpre exactamente os mesmos objectivos que no exercício anterior mas já não está em ciclo (pede só uma vez a letra e a palavra) e usa os programas *rding/rfran* e não os *ding/dfran*.

Objectivos do exercício - Redireccionamento de stdin/stdout

- Entender como funciona o redireccionamento de stdin/stdout como forma de comunicação entre processos.
- Funções *close*, *close*, *pipe*, *dup*