Exercícios
Criação e gestão de processos em Unix

Nesta aula pretende-se que os alunos fiquem com uma noção prática de criação e gestão de processos usando programação em C no sistema operativo Linux.

Exercício 1: criação de processos, uso do *fork()*

Edite e compile um programa que crie um novo processo usando o *fork()*. No processo filho, deverá escrever a *string* "Child process!" enquanto no processo pai deverá escrever a string "Parent process (child pid=%d)!", onde o *pid* do filho é obtido como resultado do *fork*.

Qual a *string* que é escrita em primeiro lugar? Tem necessariamente de ser sempre assim?

Sugestão: Se utilizar um IDE poderá não ver as mensagens do processo filho devido ao IDE fechar a consola quando o programa termina. Nesse caso, experimente usar o comando `sleep(int secs)` para obrigar o programa a esperar um pouco. Se executar o programa na consola não terá esse problema.

```
#include <stdio.h>
#include <unistd.h>

main()
{
    int pid = fork();
    if (pid == 0) {
        /* Código do processo filho */
    } else {
        /* Código do processo pai */
    }

    /* Instruções seguintes */
}
```

Exercício 2: criação de processos, uso do *execl()*

Edite e compile um programa que crie um novo processo usando o *fork.* Este deverá escrever a *string* "Child process!" no processo filho e depois, usando o comando `execl()` deverá executar o programa `ps` com um argumento à sua escolha. No processo pai deverá escrever a *string* "Parent process!".

Exercício 3: gestão de processos, uso do *wait()* e *exit()*

Edite e compile um programa que crie um novo processo usando o *fork*. No processo filho, deverá escrever a *string* "Child process (PID=%d, PPID=%d)", onde *PID* é o *process ID (pid)* do processo filho e *ppid* é o *pid* do processo pai. No processo pai deverá escrever a *string* "Parent process (PID=%d, CPID=%d)", onde *PID* é o do processo pai e *CPID* é o *pid* do processo filho.

No processo filho, após escrever a *string* anterior, deverá fazer uma pausa de 2 segundos, escrever "Child PID=%d exiting!" e usar a função `exit()` para terminar o processo.

No processo pai, deverá esperar pelo fim do processo filho usando a função `wait()`. Após receber o fim do processo filho, deverá escrever a *string* "A child has exited: CPID=%d", onde *CPID* é o *pid* do processo filho obtido como retorno da função `wait()`.

O resultado deverá ser algo semelhante ao seguinte:

```
Parent process (PID=13324, CPID=13325)
Child process (PID=13325, PPID=13324)
Child PID=13325 exiting
A child has exited: CPID=13325
```

Sugestões: pesquise pelos comando *int getpid()* e *int getppid()*.

Exercício 4: sincronização básica de processos

Edite e compile um programa que crie *n* novos processos usando o *fork*. Após a criação, cada processo filho deverá escrever a *string* "Child process (PID=%d, PPID=%d)", onde *PID* é o *process ID (pid)* do processo filho e *ppid* é o *pid* do processo pai.

Após a criação de todos os processos filhos, o processo pai deverá escrever a *string* "Parent process (PID:%d)", com o seu *process ID*.

Por fim, o processo pai deverá esperar pelo fim de todos os processos filho (usando o *wait*) e deverá listar o *pid* do processo filho que terminou.

Sugestões: note que a ordem de terminação dos processos filho pode não ser a mesma que a ordem de criação.