**As instruções fork e shmget**

Esta instrução cria um processo paralelo ao processo que chamou a fork. O processo criado, chamado processo-filho, herda, replicando, todo o ambiente que está sendo processado pelo processo-pai no momento. A partir do ponto de chamada da fork, pode-se criar códigos diferenciados para os processos filho e pai.

A sintaxe é:

**pid\_t pid = fork();**

O tipo de retorno é um identificador do processo. Em um bloco if, logo após a chamada fork, pode-se identificar os ambientes pai e filho:

**if (pid == 0) {** //\* neste bloco tem-se o ambiente filho

**... instruções ...**

**}**

**else {** //\* aqui é o ambiente pai

**... instruções ...**

**}**

No entanto, os dados do pai são herdados pelo filho em réplica, ou seja, não são compartilhados. As alterações nas variáveis feitas por um, não se refletem no ambiente do outro.

Para que se possa acessar um mesmo conjunto de dados, pode-se utilizar a instrução shmget. Esta instrução solicita a alocação de um segmento de memória externo ao processo, de forma que outros processos possam acessá-lo também.

A sintaxe é a seguinte:

**int tamanho = 5000;**

**int shmId = shmget(IPC\_PRIVATE, tamanho\*sizeof(int), IPC\_CREAT | 0666);**

**if (shmId < 0){ ... ocorreu algum erro ... }**

A instrução vai requisitar um segmento de valores int com a quantidade especificada em "tamanho".

As chaves **IPC\_XXXXX** podem ser vistas em http://man7.org/linux/man‑pages/man2/shmget.2.html

Caso a criação do segmento não tenha sido bem sucedida, haverá um retorno menor que zero.

Para que o segmento possa ser utilizado, deverá ser "atachado" ao processo. Para tanto, utiliza-se a instrução shmat:

**int \* shmPtr;**

**shmPtr = (int \*) shmat(shmId, NULL, 0);**

O ponteiro shmPtr opera como um vetor. O primeiro elemento int do segmento é acessado por shmPtr[0].

Sendo a criação e o atachamento do segmento feitos no processo pai (ainda sem a criação do filho), uma vez criado, o filho herda este ambiente, porém sem réplica, e sim compartilhado.

Programa exemplo:

**/\***

**Para executar em ambiente Unix-like.**

**By prof. Marcio Feitosa**

**\*/**

**#include <iostream>**

**#include <sys/shm.h>**

**#include <sys/wait.h>**

**#include <unistd.h>**

**#include <cmath>**

**#define TAM 1000000 // total de elementos em cada um dos vetores**

**#define VAL 200 // resultado (limite) do calculo**

**using namespace std;**

**int main()**

**{**

**cout << "-------------------------------------" << endl;**

**cout << "Programa para teste fork() + shmget()" << endl;**

**cout << "-------------------------------------" << endl;**

**cout << "Processo pai e processo filho dividem " << endl;**

**cout << "dois vetores com " << TAM << " elementos" << endl;**

**cout << "\"populados\" com inteiros entre 1 e 10" << endl;**

**cout << "-------------------------------------" << endl;**

**int shmId;**

**int \*shmPtr1, \*shmPtr2;**

**pid\_t pid;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// VETOR 1**

**// criar espaço para "TAM" elementos do tipo int**

**//**

**shmId = shmget(IPC\_PRIVATE, TAM\*sizeof(int), IPC\_CREAT | 0666);**

**if (shmId < 0)**

**{**

**cout << "Erro na criação da memória compartilhada 1";**

**return -1;**

**}**

**cout << "Espaço 1 criado com sucesso" << endl;**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// "atachar" a shared memory 1 ao processo base**

**//**

**shmPtr1 = (int \*) shmat(shmId, NULL, 0);**

**if ((int) \*shmPtr1 == -1)**

**{**

**cout << "Erro no attach 1";**

**return -1;**

**}**

**cout << "Attached 1 ok." << endl;**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// VETOR 2**

**// criar espaço para "TAM" elementos do tipo int**

**//**

**shmId = shmget(IPC\_PRIVATE, TAM\*sizeof(int), IPC\_CREAT | 0666);**

**if (shmId < 0)**

**{**

**cout << "Erro na criação da memória compartilhada 2";**

**return -1;**

**}**

**cout << "Espaço 2 criado com sucesso" << endl;**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// "atachar" a shared memory 2 ao processo base**

**//**

**shmPtr2 = (int \*) shmat(shmId, NULL, 0);**

**if ((int) \*shmPtr1 == -1)**

**{**

**cout << "Erro no attach 2";**

**return -1;**

**}**

**cout << "Attached 2 ok." << endl;**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**cout << "\n\nPressione qualquer tecla+ENTER para iniciar ";**

**char c;**

**cin >> c;**

**clock\_t inicio, fim;**

**inicio = clock();**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// "popular" o vetor com numeros aleatorios no intervalo de 1 a 10**

**//**

**srand(time(NULL));**

**for (int i = 0; i < TAM; i++)**

**{**

**shmPtr1[i] = 1 + rand() % 10;**

**shmPtr2[i] = 1 + rand() % 10;**

**}**

**//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**fim = clock();**

**double tempo\_total = 1000 \* (double) (fim - inicio) / CLOCKS\_PER\_SEC;**

**cout << "\n\nTempo para preencher os vetores = " << tempo\_total << " milisegundos";**

**cout << "\n\n\*\*\*\*\*\*\*\*\*\*\*\*\* EXECUCAO EM UM UNICO PROCESSO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;**

**inicio = clock();**

**int total = 0;**

**for (int i = 0; i < TAM; i++)**

**{**

**if (pow(shmPtr2[i], 2) + pow(shmPtr2[i], 3) > VAL)**

**{**

**total++;**

**}**

**}**

**fim = clock();**

**tempo\_total = 1000 \* (double) (fim - inicio) / CLOCKS\_PER\_SEC;**

**int tempo\_total\_um\_processo = tempo\_total;**

**cout << "Total de maiores que " << VAL << " = " << total << " (vetor1^2 + vetor2^3)";**

**cout << "\nTempo total de calculo = " << tempo\_total << " milisegundos";**

**cout << endl;**

**cout << "\n\*\*\*\*\*\*\*\*\*\*\* MESMA EXECUCAO, POREM EM 2 PROCESSOS PARALELOS \*\*\*\*\*\*\*\*\*" << endl;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// FAZER O MESMO PROCEDIMENTO COM 2 PROCESSOS**

**// pai contabiliza primeira metade do vetor e filha contabiliza a segunda**

**//**

**inicio = clock(); // marcar tempo de inicio**

**pid = fork();**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// processo filho**

**//**

**if (pid == 0)**

**{**

**int total = 0;**

**for (int i = TAM/2; i < TAM; i++) // corre da metade ao fim do vetor**

**{**

**if (pow(shmPtr2[i], 2) + pow(shmPtr2[i], 3) > VAL)**

**{**

**total++;**

**}**

**}**

**cout << "Total filho = " << total << endl;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// processo pai**

**//**

**else**

**{**

**int total = 0;**

**for (int i = 0; i < TAM/2; i++) // corre do inicio `a metade do vetor**

**{**

**if (pow(shmPtr2[i], 2) + pow(shmPtr2[i], 3) > VAL)**

**{**

**total++;**

**}**

**}**

**cout << "Total pai = " << total << endl;**

**wait(NULL); // aguarda o termino de todos os filhos**

**fim = clock();**

**tempo\_total = 1000 \* (double) (fim - inicio) / CLOCKS\_PER\_SEC;**

**cout << "\nTempo total de calculo (pai + filho) = " << tempo\_total << " milisegundos" << endl;**

**cout << "SpeedUp = " << (double) tempo\_total\_um\_processo / tempo\_total << endl;**

**cout << "\*\*\* FIM \*\*\*" << endl << endl;**

**}**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**return 0;**

**}**