Ficha Prática 3

Sistemas Operativos 2023

Processos e I/O Baixo Nível:

Título : Servidor de Inteiros

Capítulos 7 e 8 das Aulas Praticas : https://www.di.ubi.pt/~operativos/praticos/praticos.html

A funcionalidade do mecanismo fork() é criar novos processos. Por exemplo: um interpretador de linha de comandos, como a Bash Shell, lança um programa novo a partir da leitura de uma string; num interpretador gráfico como o explorador é detectado um clique e será aberto um programa novo; em ambos os casos através dum mecanismo tipo fork-exec ou spawn. Outra grande utilidade do fork() diz respeito aos programas ditos servidores onde um pedido dum serviço é feito num processo novo criado (*forked*) a partir do processo principal, um exemplo é um servidor de ssh
Neste exercício pretende-se criar um servidor de números inteiros. O programa, deve receber como parâmetro uma string. Depois num processo novo (filho) deve ler de um ficheiro (binário) os números pretendidos, para de seguida visualizar no ecrã. Conforme o input fornecido ao servidor o processo criador espera ou não a terminação do processo filho.

1 Qual é o output do seguinte programa?

Deverá mostrar o funcionamento do programa como fluxograma (como foi feito nas aulas teóricas) e verifcar compilando e executando o programa.

```
int pid, x = 3;
pid = fork();
if (0 == pid)
{
   fork();
   pid = fork();
   x--;
   if (0 == pid) x--;
} else
      x++;
printf("x=%d\n ",x);
```

2 qual é o output do seguinte programa?

O Output do programa seguinte é “Determinístico” i.e o output é sempre pelo mesmo ordem. Considerando que quando for executado o primeiro processo a ser criado terá o PID=3000 e todos os próximos processos que serão criados terão o próximo valor sequencial do PID disponível (3001,3002 ETC)

Faça um diagrama temporal dos processos (trace) indicando em cada ponto quando uma das instruções é executado (as instruções (i1,i2,..i6) que forem etiquetadas.) Deverá indicar os valores das variáveis importantes.
Indicar o output do programa. Verificar a sua resposta compilando e executando programa.

```
int main ()
{
     int pid1, pid2, pid3, pid4;

i1:  pid1 = getpid ();          /* obtain the current PID   */

i2:  pid2 = fork ();            /* fork parent and child    */
i3:  wait (NULL);               /* parent waits for its child */

i4:  pid3 = fork ();
i4:  wait (NULL);               /* parent waits for its child */

i5:  pid4 = getpid ();
i6:  printf ("%4d \t %4d \t %4d \t %4d\n", pid1, pid2, pid3, pid4);
}
```

3a – Criar um ficheiro binário com números inteiros

```
int main() {
  srand(xxxx);  //xxxx no aluno
  int vetor[100], inicio=rand();
  for ( int i=0; i <100 ; i++ )  vetor[i]=inicio+i;
  int fd = creat ( "ints.bin", permissões );
  write(fd, vetor, 100*sizeof(int) );
}
```

3b Escrever o programa – servidor.c.

O programa servidor deve aceitar strings apenas com os seguintes formatos:

EX 0 0
NG A B fileName
NE A B

O processo filho criado com fork() abre o ficheiro binário criado em cima de números inteiros efectuará a leitura dos números nas posições entre A e B (A e B são inteiros com 0<=A<B<=99) e a seguir mostra-os no ecrã ou escrver-los para um ficheiro.

Algoritmo para Implementar

Num Ciclo infinito

- Ler uma linha do ecrã
- Conforme os valores lidos :
  - Se o valor da string é igual à string "EX" então o programa deverá terminar
  - Se o valor da string é "NG" então o processo servidor criará o processo filho e continuará a sua execução
  - Se o valor da string é "NE" então o processo servidor criará o processo filho e espera a sua terminação (precisará de utilizar a função wait())
  - Caso contrário imprime-se a mensagem "Protocolo Desconhecido"
- O processo filho abrirá o ficheiro de números inteiros e mostra os que estão dentro do intervalo especificado no ecrã ou num ficheiro.

Encerramento

Alterar o nome do ficheiro servidor.c para axxxxx.c em que axxxx representa o seu número de aluno e entregar no moodle.

Bibliotecas Padrão

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
```

Leitura das Variaveis

```
{
        char linha[200];
        char s1[3]="", s2[100]="";
        int a, b;
        fgets(linha, 200, stdin);
        sscanf(linha, "%s %d %d %s", s1, &a, &b, s2);
        printf("%s %d %d %s\n", s1, a, b, s2);
}
```

Modos de acesso a Ficheiros `#define FILE_MODE (S_IRUSR | S_IWUSR)`
Funções Uteis: **`read, write, open, creat, lseek, scanf, print, strcmp, fork, wait`**

Para ver o conteúdo dum ficheiro binário poderá usar o utilizar o comando linux "**`od`**"