Laboratório de Fork e Pid

Sistemas de Computação - 3WB

Victor Augusto - 1310784

ÍNDICE

1. Introdução

Foram propostos 2 exercícios em aula. No primeiro era pedido para executar o somatório de matrizes em pelo menos 3 processos. No segundo era necessário executar a busca de palavras em um poema, sendo um processo diferente para cada palavra.

2. Objetivo

Reforçar os conceitos sobre PID, Fork e os assuntos associados.

3. Exercício 1

3.1 Enunciado

Elaborar um programa que leia os elementos de duas matrizes 3 X 3 de um arquivo texto contendo: número da linha, número da coluna, valor, e que realize a soma das matrizes lidas da seguinte forma:

- a soma dos elementos deve ser feita por, no mínimo, 3 processos.
- a comunicação entre os processos deve ser feita via arquivo.
- os arquivos são compartilhados entre pai e filho.

3.2 Código Fonte

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>


#define NUM_FILHOS 3


int main (void) {


        pid_t pid, pid1, pid2, pid3;
```

```
int status;
int linha, coluna, valor = 0;
FILE *fileMatriz1;
FILE *fileMatriz2;
FILE *fileMatrizResultado;
FILE *fileMatrizResultadoRead;
int matriz1[3][3];
int matriz2[3][3];
int matrizResultado[3][3];

fileMatriz1 = fopen("fileMatriz1.txt", "r");
if (fileMatriz1 == NULL) {
        printf("Error opening file!\n");
        exit(1);
}

fileMatriz2 = fopen("fileMatriz2.txt", "r");
if (fileMatriz1 == NULL) {
        printf("Error opening file!\n");
        exit(1);
}

fileMatrizResultado = fopen("fileMatrizResultado.txt", "w");
if (fileMatrizResultado == NULL) {
        printf("Error opening file!\n");
        exit(1);
}

fileMatrizResultadoRead = fopen("fileMatrizResultado.txt", "r");
if (fileMatrizResultado == NULL) {
        printf("Error opening file!\n");
        exit(1);
```

```
}

printf("Matriz 1:\n");
while(fscanf(fileMatriz1, "%d %d %d", &linha, &coluna, &valor) == 3) {
        matriz1[linha][coluna] = valor;
        printf("%d %d %d\n",linha, coluna, valor);
}
fclose(fileMatriz1);

printf("\nMatriz 2:\n");
while(fscanf(fileMatriz2, "%d %d %d", &linha, &coluna, &valor) == 3) {
        matriz2[linha][coluna] = valor;
        printf("%d %d %d\n",linha, coluna, valor);
}
fclose(fileMatriz2);

printf("\nMatriz 1:\n");
for(int x = 0; x<3; x++) {
        printf("%d %d %d\n", matriz1[x][0], matriz1[x][1], matriz1[x][2]);
}

printf("\nMatriz 2:\n");
for(int x = 0; x<3; x++) {
        printf("%d %d %d\n", matriz2[x][0], matriz2[x][1], matriz2[x][2]);
}

pid = 1;

printf("\nProcesso Pai Inicio\n”);

for(int j = 0; j<NUM_FILHOS; j++) {
        pid = fork();
```

```
            if (pid == 0) {
                printf("\nProcesso Filho %d Inicio\n\n", j+1);
                for (int i = 0; i < 3; i++) {
                    int soma = matriz1[j][i] + matriz2[j][i];
                    fprintf(fileMatrizResultado, "%d %d %d\n", j, i, soma);
                    printf("Linha: %d / Coluna: %d / Valor: %d\n", j, i, soma);
                }
                printf("\nProcesso Filho %d Fim\n\n", j+1);
                exit(0);
            }
        }

        for (int i = 0; i < NUM_FILHOS; i++) {
            waitpid(-1, &status, 0);
        }

        printf("\nProcesso Pai Continua\n");

        printf("\nMatriz Resultado:\n");
        while(fscanf(fileMatrizResultadoRead, "%d %d %d", &linha, &coluna, &valor) == 3) {
            matrizResultado[linha][coluna] = valor;
            printf("%d %d %d\n",linha, coluna, valor);
        }
        fclose(fileMatrizResultadoRead);
        fclose(fileMatrizResultado);

        printf("\nMatriz Resultado:\n");
        for(int x = 0; x<3; x++) {
            printf("%d %d %d\n", matrizResultado[x][0], matrizResultado[x][1],
matrizResultado[x][2]);
        }
```

```
        printf("\nProcesso Pai Fim\n\n");


        return 0;
}
```

3.3 Resultado

Matriz 1:

0 0 1

0 1 2

0 2 3

1 0 1

1 1 2

1 2 3

2 0 1

2 1 2

2 2 3

Matriz 2:

0 0 1

0 1 2

0 2 3

1 0 1

1 1 2

1 2 3

2 0 1

2 1 2

2 2 3

Matriz 1:

1 2 3

1 2 3

1 2 3

Matriz 2:

1 2 3

1 2 3

1 2 3

printf("\nProcesso Pai Inicio\n”);

Processo Filho 1 Inicio

Linha: 0 / Coluna: 0 / Valor: 2

Linha: 0 / Coluna: 1 / Valor: 4

Linha: 0 / Coluna: 2 / Valor: 6

Processo Filho 1 Fim

Processo Filho 2 Inicio

Processo Filho 3 Inicio

Linha: 1 / Coluna: 0 / Valor: 2

Linha: 1 / Coluna: 1 / Valor: 4

Linha: 1 / Coluna: 2 / Valor: 6

Processo Filho 2 Fim

Linha: 2 / Coluna: 0 / Valor: 2

Linha: 2 / Coluna: 1 / Valor: 4

Linha: 2 / Coluna: 2 / Valor: 6

Processo Filho 3 Fim

Processo Pai Continua

Matriz Resultado:

0 0 2

0 1 4

0 2 6

1 0 2

1 1 4

1 2 6

2 0 2

2 1 4

2 2 6

Matriz Resultado:

2 4 6

2 4 6

2 4 6

Processo Pai Fim

4. Exercício 2

4.1 Enunciado

Elaborar um programa que procure as seguintes palavras em um arquivo texto contendo o poema “Irene no céu” de Manoel Bandeira:
“Irene”, “ceu”, “humor”, “Pedro”, “boa”.

Cada trabalhador buscará uma palavra e indicará em um arquivo a sua palavra e o número de ocorrências. Ao final o programa deverá listar as palavras e o numero de ocorrências de cada uma.

4.2 Código Fonte

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

#define NUM_FILHOS 5

int searchWord(char* word);

int main(void) {

   pid_t pid;

   FILE* words;

   char* search[5] = {"Irene", "ceu", "humor", "Pedro", "boa"};

   words = fopen("words.txt", "w");
   if (words == NULL) {
      printf("Error opening file!\n");
                exit(1);
   }

   printf("Começo do Processo Pai\n");

   pid = 1;

         for(int j = 0; j<NUM_FILHOS; j++) {
                pid = fork();
                if (pid == 0) {
                        printf("\nComeço do Processo Filho %d\n\n", j+1);
         int count = searchWord(search[j]);
         fprintf(words, "%s %d\n",search[j], count);
                        printf("\nFim do Processo Filho %d\n\n", j+1);
                        exit(0);
                }
         }

   int i = 0;
   while (i < 5) {
      waitpid(-1, NULL, 0);
      i++;
   }

   printf("\nContinua do Processo Pai\n\n");

   fclose(words);

   words = fopen("words.txt", "r");
   if (words == NULL) {
      printf("Error opening file!\n");
                exit(1);
   }

   char word[10];
   int count;

   while(fscanf(words, "%s %d", word, &count) != EOF) {
```

```
        printf("%s %d\n", word, count);
    }

    fclose(words);
    printf("\nFim do Processo Pai\n\n");

    return 0;
}

int searchWord(char* word) {
    FILE* poema;
    char find[100];
    int value = 0;
    poema = fopen("poema.txt", "r");
    while (fscanf(poema, "%s ", find) != EOF) {
        if (strstr(find, word) != NULL) {
            value++;
        }
    }
    fclose(poema);
    return value;
}
```

4.3 Resultado

```
Começo do Processo Pai



Começo do Processo Filho 1



Procurando: Irene



Começo do Processo Filho 2



Procurando: ceu



Fim do Processo Filho 1









Começo do Processo Filho 3

Fim do Processo Filho 2
```

Procurando: humor

Fim do Processo Filho 3

Começo do Processo Filho 4

Procurando: Pedro

Fim do Processo Filho 4

Começo do Processo Filho 5

Procurando: boa

Fim do Processo Filho 5

Continua do Processo Pai

Irene 6

ceu 1

humor 1

Pedro 1

boa 1

Fim do Processo Pai

5. Conclusões

5.1 Exercício 1

Com a saída do exercício 1 percebe-se que as somas são executadas independentemente e a escrita da matriz é na ordem que os processos terminam. Cabe ao processo pai esperar que todas acabem e apresentar a matriz na de forma correta.

5.2 Exercício 2

Pela saída apresentada é notável que as pesquisas estão sendo executadas simultaneamente em processos separados, e a ordem de escrita do resultado é igual a ordem de finalização dos processos.

Para resolver esse exercício utilizei uma função de string que nunca tinha usado antes: strstr. Dessa forma eu procuro a string que eu quero dentro de outra, o que resolve o caso de pesquisa pela palavra “Irene” no momento que ela aparece no texto com “Irene.”, pois a palavra está contida na string, mesmo que tenha um “.” no final.

5.3 Gerais

Vale dizer que até os exercícios em questão, acreditava que waitpid(-1, NULL, 0) esperava todos os processos filhos acabarem. Depois de ler o manual entendi que eles espera algum processo filho acabar para depois continuar, sendo assim necessário dar waitpid(-1, NULL, 0) para todos os processos filhos.

Além disso os exercícios foram essenciais para entender e aprender a gerenciar diversos processos filhos da melhor maneira possível, sem deixar o código muito extenso.

Outro importante aprendizado foi a troca de informações entre processos através de arquivos, que até então não tinha sido feito no laboratório.