**UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP**

**CIÊNCIA DA COMPUTAÇÃO**

****

****

**SISTEMAS OPERACIONAIS**

**TRABALHO PRÁTICO 2 – PT2**

Marcus Vinícius Souza Fernandes

19.1.4046

**Ouro Preto**

**2021**

**Código 01**

O código a seguir utiliza 3 forks e com isso nos gera três processos (impressão, incrementação e decrementarão). Toda a interação é feita através da leitura instantânea do terminal. (Anotações nos comentários)

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <stdbool.h>

#include <unistd.h>

#include <sys/wait.h>

#include <signal.h>

#define READ  0

#define WRITE 1

*void* interface();

*int* main(){

    //Chamando função de interface de interação

    interface();

    //Criando pipes e forks e verificando sucesso.

*int* saldo[2];

*int* opcao[2];

*int* caractere = 0;

    if (pipe(saldo) < 0 || pipe(opcao) < 0){

        printf("Erro ao criar pipe.\n");

        exit(1);

    }

*int* aux = 0;

    if (write(saldo[WRITE], &aux, sizeof(*int*)) == -1){

        printf("Erro ao escrever no pipe.\n");

        exit(1);

    }

*pid\_t* pidPai = getpid();

*pid\_t* filho1 = fork();

    if (filho1 < 0){

        printf("Erro ao criar fork.\n");

        exit(1);

    }

    fflush(stdin);

*pid\_t* filho2 = -1;

    if (getpid() == pidPai){

        filho2 = fork();

        if (filho2 < 0){

            exit(1);

        }

        if (filho2 == 0){

            filho1 = -1;

        }

    }

    if (getpid() == pidPai){

*int* x;

        // K & k (Finalizar execução)

        while(caractere != 107 || caractere != 75){

            // + S s k K -

            do{

                caractere = (*int*)getchar();

            }while((caractere != 43) && (caractere != 83) && (caractere != 115) && (caractere != 107) && (caractere != 75) && (caractere != 45));

            // S s

            if (caractere == 83 || caractere == 115){

                read(saldo[READ], &x, sizeof(*int*));

                printf("-----------------------------\n");

                printf("Print em PID : %d\n", getpid());

                printf("Saldo        : %d\n", x);

                printf("-----------------------------\n");

                write(saldo[WRITE], &x, sizeof(*int*));

            }

            else{

                write(opcao[WRITE], &caractere, sizeof(*int*));

            }

            // k K

            if (caractere == 107 ||caractere == 75){

                printf("-----------------------------\n");

                kill(filho1, SIGKILL);

                kill(filho2, SIGKILL);

                kill(getpid(), SIGKILL);

                exit(0);

            }

        }

    }

    else if (filho1 == 0){

*int* operacao;

*int* x;

        do{

            read(opcao[READ], &operacao, sizeof(*int*));

            // +

            if (operacao == 43){

                printf("-----------------------------\n");

                printf("Soma em PID  : %d\n", getpid());

                printf("-----------------------------\n");

                read(saldo[READ], &x, sizeof(*int*));

                x += 100;

                write(saldo[WRITE], &x, sizeof(*int*));

            }

            else{

                write(opcao[WRITE], &operacao, sizeof(*int*));

            }

        // k K

        }while(operacao != 107 ||operacao != 75 );

    }

    else if (filho2 == 0){

*int* operacao;

*int* x;

        do{

            read(opcao[READ], &operacao, sizeof(*int*));

            // -

            if(operacao == 45){

                printf("-----------------------------\n");

                printf("Remove em PID: %d\n", getpid());

                printf("-----------------------------\n");

                read(saldo[READ], &x, sizeof(*int*));

                x -= 50;

                write(saldo[WRITE], &x, sizeof(*int*));

            }

            else{

                write(opcao[WRITE], &operacao, sizeof(*int*));

            }

        // k K

        }while(operacao != 107 ||operacao != 75);

    }

    else{}

    return 0;

}

*void* interface(){

    printf("\n");

    printf("Pressione [+] para somar 100 UD\n");

    printf("Pressione [-] para subtrair 50 UD\n");

    printf("Pressione [s] para imprimir o valor do saldo\n");

    printf("Pressione [k] para finalizar a execucao\n");

    printf("Pressione [ENTER] para confirmar as operacoes\n\n");

    printf("Valor inicial de saldo: 0\n");

}

**Código 02**

O código a seguir parte do mesmo principio do mesmo visto acima, porém desta vez com a utilização de threads. (Anotações nos comentários)

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <sys/types.h>

#include <unistd.h>

#include <signal.h>

#include <sys/syscall.h>

*int* SALDO = 0;

*int* KEY;

*struct* *T\_args* {

*pthread\_t* thread;

};

*void* interface();

// Funções de cada uma das threads

*void* \*addMoneyUnit(*void* \**arg*);

*void* \*removeMoneyUnit(*void* \**arg*);

*void* \*printMoneyUnit(*void* \**arg*);

*void* \*restartSystem(*void* \**arg*);

*int* main(){

    interface();

*pthread\_t* addMoneyUnitThread;

*pthread\_t* removeMoneyUnitThread;

*pthread\_t* printMoneyUnitThread;

*pthread\_t* restartSystemThread;

    // Captura do valor da tecla sem pressionar ENTER

    system("/bin/stty raw");

    // Argumentos das threads

*struct* *T\_args* \*threadPrint  = (*struct* *T\_args* \*) malloc(sizeof(*struct* *T\_args*));

*struct* *T\_args* \*threadAdd    = (*struct* *T\_args* \*) malloc(sizeof(*struct* *T\_args*));

*struct* *T\_args* \*threadRemove = (*struct* *T\_args* \*) malloc(sizeof(*struct* *T\_args*));

    threadPrint->thread  = printMoneyUnitThread;

    threadAdd->thread    = addMoneyUnitThread;

    threadRemove->thread = removeMoneyUnitThread;

    // Cria threads de soma, remoção e impressão

    pthread\_create(&printMoneyUnitThread,  NULL, printMoneyUnit,  (*void* \*) threadPrint);

    pthread\_create(&addMoneyUnitThread,    NULL, addMoneyUnit,    (*void* \*) threadAdd);

    pthread\_create(&removeMoneyUnitThread, NULL, removeMoneyUnit, (*void* \*) threadRemove);

    do{

        // Altera opções

        KEY = getc(stdin);

        // Mata threads

        // k

        if (KEY == 107){

            pthread\_kill(addMoneyUnitThread, 0);

            pthread\_kill(removeMoneyUnitThread, 0);

            pthread\_kill(printMoneyUnitThread, 0);

            printf("\n=======================================\n");

            printf("Terminando (kill) Threads... OK\n");

            printf("=======================================\n");

        }

        // Reinicia o sistema limpando operações e tela

        // r

        else if(KEY == 114){

            // Termina as thread em execução e cria novas

            pthread\_join(addMoneyUnitThread, NULL);

            pthread\_kill(addMoneyUnitThread, 0);

            pthread\_join(removeMoneyUnitThread, NULL);

            pthread\_kill(removeMoneyUnitThread, 0);

            pthread\_join(printMoneyUnitThread, NULL);

            pthread\_kill(printMoneyUnitThread, 0);

            pthread\_create(&restartSystemThread, NULL, restartSystem, NULL);

            pthread\_create(&printMoneyUnitThread, NULL, printMoneyUnit, (*void* \*) threadPrint);

            pthread\_create(&addMoneyUnitThread, NULL, addMoneyUnit, (*void* \*) threadAdd);

            pthread\_create(&removeMoneyUnitThread, NULL, removeMoneyUnit, (*void* \*) threadRemove);

            pthread\_join(restartSystemThread, NULL);

            interface();

        }

    // k

    }while(KEY != 107);

    system ("/bin/stty cooked");

    return  0;

}

// Interface do menu

*void* interface(){

    printf("--------------------------------------------\n");

    printf("Pressione [+] Para adicionar 100 UD ao saldo\n");

    printf("Pressione [-] Para retirar 50 UD do saldo   \n");

    printf("Pressione [s] Para imprimir o saldo         \n");

    printf("Pressione [k] Para matar as thread criadas  \n");

    printf("Pressione [r] Para limpar a tela e operações\n");

    printf("--------------------------------------------\n\n");

}

// Deposita 100 unidadeds de dinheiro

*void* \*addMoneyUnit(*void* \**arg*){

    do{

        if (KEY == 43){

            printf("\nAdicionando 100 UD com TID: %ld\n", syscall(\_\_NR\_gettid));

            SALDO += 100;

            KEY = 0;

        }

        else if (KEY == 107 || KEY == 114){

            pthread\_exit(0);

        }

    }while(1);

}

// Retira 50 unidadeds de dinheiro

*void* \*removeMoneyUnit(*void* \**arg*){

    do{

        if (KEY == 45){

            printf("\nRemovendo 50 UD com TID   : %ld\n", syscall(\_\_NR\_gettid));

            SALDO -= 50;

            KEY = 0;

        }

        else if (KEY == 107 || KEY == 114){

            pthread\_exit(0);

        }

    }while(1);

}

// Printa saldo

*void* \*printMoneyUnit(*void* \**arg*){

    do{

        if (KEY == 115){

            printf("\n----------------------------------------\n");

            printf("Mostrando saldo UD com TID: %ld\n", syscall(\_\_NR\_gettid));

            printf("Saldo em conta            : %d UD\n", SALDO);

            printf("----------------------------------------\n");

            KEY = 0;

        }

        else if (KEY == 107 || KEY == 114){

            pthread\_exit(0);

        }

    }while(1);

}

// Reinicia

*void* \*restartSystem(*void* \**arg*){

    SALDO = 0;

    KEY = 0;

    system("clear");

    pthread\_exit(0);

}