Universidade Presbiteriana Mackenzie  
Ciência da Computação  
Sistemas Operacionais – Turma 04P11  
Alan Meniuk Gleizer – 10416804  
Caio Vinicius Corsini Filho – 10342005

Relatório Lab 03 – Criação de Processos

Exercício 01  
código fonte

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> // Para fork() e sleep()

#include <sys/types.h> // Para pid\_t

int main() {

    int pid;

    pid = fork(); // Cria processo filho

    // No processo pai, retorna o PID do processo filho criado

    // No processo filho, retorna 0

    // Erro: retorna -1 (e nenhum processo é criado)

    if (pid < 0) {

        printf("Erro na criação do filho\n");

        return 1;

    }

    if (pid == 0) {

        // se estamos no processo filho

        // imprimir PID do filho e valor da variável pid

        printf("Processo Filho: PID = %d, valor do PID (var) = %d.\n", getpid(), pid);

        // Loop (5x) do processo filho

        for (int i = 0; i < 5; i++) {

            printf("Mensagem %d - processo filho\n", i + 1);

            sleep(1); // Espera por 1 segundo

        }

    } else {

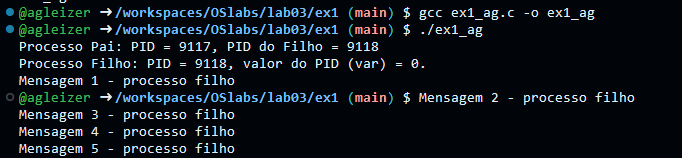
        // se pid > 0, imprimimos PID do processo pai

        printf("Processo Pai: PID = %d, PID do Filho = %d\n", getpid(), pid);

    }

    return 0;

}

print execução

Exercício 02  
código fonte

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> // Para fork() e sleep()

#include <sys/types.h> // Para pid\_t, não usado

#include <sys/wait.h> // Para wait()

int main() {

    int pid;

    pid = fork(); // Cria processo filho

    // No processo pai, retorna o PID do processo filho criado

    // No processo filho, retorna 0

    // Erro: retorna -1 (e nenhum processo é criado)

    if (pid < 0) {

        printf("Erro na criação do filho\n");

        return 1;

    }

    if (pid == 0) {

        // se estamos no processo filho

        // imprimir PID do filho e valor da variável pid

        printf("Processo Filho: PID = %d, valor do PID (var) = %d.\n", getpid(), pid);

        // Loop (5x) do processo filho

        for (int i = 0; i < 5; i++) {

            printf("Mensagem %d - processo filho\n", i + 1);

            sleep(1); // Espera por 1 segundo

        }

    } else {

        // se pid > 0, imprimimos PID do processo pai

        printf("Processo Pai: PID = %d, PID do Filho = %d\n", getpid(), pid);

        wait(NULL);

        // Espera o término do processo filho

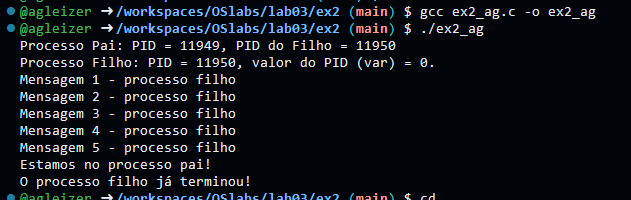
        // fonte: https://stackoverflow.com/questions/42426816/in-what-way-does-waitnull-work-exactly-in-c

        printf("Estamos no processo pai!\nO processo filho já terminou!\n");

    }

    return 0;

}

print execução

Exercício 3  
código fonte

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> // Para fork(), exec() e sleep()

#include <sys/types.h> // Para pid\_t, não usado

#include <sys/wait.h> // Para wait()

int main() {

    int pid;

    pid = fork(); // Cria processo filho

    // No processo pai, retorna o PID do processo filho criado

    // No processo filho, retorna 0

    // Erro: retorna -1 (e nenhum processo é criado)

    if (pid < 0) {

        printf("Erro na criação do filho\n");

        return 1;

    }

    if (pid == 0) {

        // se estamos no processo filho

        // imprimir PID do filho e valor da variável pid

        printf("Processo Filho: PID = %d, valor do PID (var) = %d.\n", getpid(), pid);

        // Substitui o processo filho pelo comando 'ls -l'

        execlp("ls", "ls", "-l", NULL);

        // erro se exec() falhar

        printf("Falha ao executar execlp\n");

        return 1;

    } else {

        // Código do processo pai

        printf("Processo Pai: PID = %d, PID do Filho = %d\n", getpid(), pid);

        // Espera o término do processo filho

        wait(NULL);

        printf("Estamos no processo pai!\nO processo filho já terminou!\n");

    }

    return 0;

}

print execução

