Pontifícia Universidade Católica do Rio de Janeiro

Lab 4 - Sinais

Sistemas Operacionais

Alunos Tharik Lourenço
Suemy Inagaki

Professor Luiz Fernando Seibel

Rio de Janeiro, 01 de maio de 2021

Conteúdo

1 Problema 1

Elabore um programa que simule um escalonador com tempo compartilhado. Suponha que 4 processos chegam na fila de prontos no instante t0 (processos p1, p2, p3 e p4), nesta ordem. O Programa deve criar 4 processos filhos e alternar a execução dos filhos, a cada segundo, usando sinais. Após 20 trocas de contexto de um processo filho, o processo pai envia um sinal a esse filho para que ele termine. Isso acontece com os 4 processos. Os filhos são formados por loops eternos. Somente devem ser exibidas mensagens de início de execução e término de cada processo. Explique a alternância dos processos

1.1 Código Fonte

```
/*
Tharik Lourenço
Suemy Inagaki
Lab 04 Sinais
Entrega Final
Sistemas Operacionais
*/


#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>

#define EVER ;;
#define NUMPROC 4

void trata_SIGTSTP(int signum);
void trata_SIGCONT(int signum);

void trata_SIGTSTP(int signum) {
  printf("processo %d parado.\n", getpid());
  signal(SIGCONT, trata_SIGCONT);
  signal(SIGTSTP, SIG_DFL);
  raise(SIGTSTP);
}
```

```c
void trata_SIGCONT(int signum) {
  printf("processo %d continue\n", getpid());
  signal(SIGTSTP, trata_SIGTSTP);
  signal(SIGCONT, SIG_DFL);
  raise(SIGCONT);
}


int main (int argc, char *argv[]){
    pid_t filhopid[NUMPROC];
    pid_t pid;
    for (int i=0 ;i<NUMPROC;i++){
        pid = fork();
        filhopid[i] = pid;
        signal(SIGTSTP, trata_SIGTSTP);
        signal(SIGCONT, trata_SIGCONT);
        if (pid < 0){
            fprintf(stderr, "Erro ao criar filho\n");
            exit(-1);
        }
        if (pid == 0){ /* child */
            printf("Filho %d criado pid: %d\n", i, getpid());
            for(EVER);
        }
    }
    /* parent */
    for(int j =0; j<20; j++){
        printf("tentando - %d\n", j+1);
        for(int i=0 ;i<NUMPROC ;i++){
            kill(filhopid[i], SIGCONT);
            sleep(1);
            kill(filhopid[i], SIGTSTP);
        }
    }
    for(int i=0 ;i<NUMPROC;i++)
        kill(filhopid[i], SIGKILL);

    return 0;
}
```

1.2 Resultado da Execução

```
$ gcc entrega_final.c -o entrega_final
$ ./entrega_final

Filho 0 criado pid: 518
Filho 1 criado pid: 519
Filho 2 criado pid: 520
Filho 3 criado pid: 521
tentando - 1
processo 518 continue
processo 519 continue
processo 518 parado.
processo 520 continue
processo 519 parado.
processo 521 continue
processo 520 parado.
processo 521 parado.
tentando - 2
processo 518 continue
processo 518 parado.
processo 519 continue
processo 519 parado.
processo 520 continue
processo 520 parado.
processo 521 continue
processo 521 parado.
tentando - 3
processo 518 continue
processo 518 parado.
processo 519 continue
processo 519 parado.
processo 520 continue
processo 520 parado.
processo 521 continue
processo 521 parado.
tentando - 4
processo 518 continue
processo 518 parado.
processo 519 continue
processo 519 parado.
processo 520 continue
processo 520 parado.
processo 521 continue
processo 521 parado.
tentando - 5
processo 518 continue
processo 518 parado.
processo 519 continue
processo 519 parado.
processo 520 continue
processo 520 parado.
processo 521 continue
processo 521 parado.
```

1.3 Explicação

A função trata_SIGTSTP pausa um processo quando capta o sinal SIGTSTP. A função trata_SIGCONT continua um processo quando capta o sinal SIGCONT. Criamos 4 processos filhos e os deixamos em um loop eterno. Enquanto isso, no processo pai, depois de 20 trocas de contexto de um processo filho, o pai envia um sinal e termina o filho.