LAB4 – Sinais e escalonamento de processos de tempo real

Hanna Epelboim Assunção - 2310289
Érica Oliveira Regnier - 2211893

Na política de escalonamento REAL-TIME cada processo deve executar periodicamente (uma vez por minuto), iniciando sua execução, em determinado momento de tempo (I) e deve permanecer executando apenas durante um certo período de tempo (D).
Escreva em C um programa escalonador que executa um loop infinito criando três processos filho, p1, p2 e p3, e que os escalona da seguinte maneira (a cada minuto):
• P1 inicia após 5 segundos (do início do minuto cheio) e executa por 20 segundos
• P2 inicia após 45 segundos (do minuto cheio) e executa durante 15 segundos
• P3 executa quando nem P1 e nem P2 executam
Obs: Use sinais para iniciar/continuar e interromper a execução dos processo (SIGCONT e SIGSTOP), use a chamada gettimeofday (&T,NULL) onde T é do tipo struct timeval, cujo campo tv_sec contém, o número de segundos transcorridos desde 1/1/1970

Código fonte:

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
int pid1, pid2, pid3;

void handler(int signo){
   kill(pid1,SIGKILL);
   kill(pid2,SIGKILL);
   kill(pid3,SIGKILL);
   exit(0);
}

int main(int argc, char** argv){

   signal(SIGALRM, handler);

   struct timeval tempoAtual;
```

```
int mod;
short flag0 = 1,flag1 = 1, flag2 = 1, flag3 = 1;
pid1 = fork();
if(pid1<0){
    perror("Erro no fork()");
    exit(1);
}
else if(pid1 == 0){
    execvp(argv[1],argv);
}
kill(pid1,SIGSTOP);

pid2 = fork();
if(pid2<0){
    perror("Erro no fork()");
    exit(1);
}
else if(pid2 == 0){
    execvp(argv[2],argv);
}
kill(pid2,SIGSTOP);

pid3 = fork();
if(pid3<0){
    perror("Erro no fork()");
    exit(1);
}
else if(pid3 == 0){
    execvp(argv[3],argv);
}
kill(pid3,SIGSTOP);

while(flag0){
    gettimeofday(&tempoAtual,NULL);
    mod = (tempoAtual.tv_sec)%60;
    if(mod == 0){
        flag0 = 0;
        alarm(120); //dois min
```

```
        while(1){
            gettimeofday(&tempoAtual,NULL);
            mod = (tempoAtual.tv_sec)%60;
            if(mod == 0){
                if(flag3){
                    kill(pid2,SIGSTOP);
                    kill(pid3,SIGCONT);
                    printf("Tempo: %d - Iniciando filho 3\n",mod);
                    flag3 = 0;
                    flag2 = 1;
                }
            }
            else if(mod == 25){
                if(flag3){
                    kill(pid1,SIGSTOP);
                    kill(pid3,SIGCONT);
                    printf("Tempo: %d - Iniciando filho 3\n",mod);
                    flag3 = 0;
                    flag1 = 1;
                }
            }
            else if(mod == 45){
                if(flag2){
                    kill(pid3,SIGSTOP);
                    kill(pid2,SIGCONT);
                    printf("Tempo: %d - Iniciando filho 2\n",mod);
                    flag2 = 0;
                    flag3 = 1;
                }
            }
            else if(mod == 5){
                if(flag1){
                    kill(pid3,SIGSTOP);
                    kill(pid1,SIGCONT);
                    printf("Tempo: %d - Iniciando filho 1\n",mod);
                    flag1 = 0;
                    flag3 = 1;
                }
```

```
                }
            }
        }
    }
    return 0;
}
```

Código filho.c:

```
#include <stdio.h>
#include <unistd.h>
int main(void){
    while(1){

    }
}
```

Comandos:

```
[c2310289@ciborgue l4]$ gcc -Wall -o escal l4.c
[c2310289@ciborgue l4]$ gcc -Wall -o f1 filho.c
[c2310289@ciborgue l4]$ gcc -Wall -o f2 filho.c
[c2310289@ciborgue l4]$ gcc -Wall -o f3 filho.c
[c2310289@ciborgue l4]$ ./escal ./f1 ./f2 ./f3
```

Saída:

```
Tempo: 0 - Iniciando filho 3
Tempo: 5 - Iniciando filho 1
Tempo: 25 - Iniciando filho 3
Tempo: 45 - Iniciando filho 2
Tempo: 0 - Iniciando filho 3
Tempo: 5 - Iniciando filho 1
Tempo: 25 - Iniciando filho 3
Tempo: 45 - Iniciando filho 2
[c2310289@valquiria l4]$
```

Explicação:

A função handler(int signo) é chamada quando o processo principal recebe o sinal SIGALRM. Ao ser acionada, esta função envia o sinal SIGKILL para os três processos filhos (pid1, pid2, pid3), encerrando-os, e depois chama exit(0) para finalizar o processo principal. O sinal SIGALRM é programado para ser enviado após 120 segundos (2 minutos), configurado mais adiante no código. O processo principal cria três processos filhos usando fork(), que retorna o PID do filho para o processo pai, e 0 para o próprio processo filho. Se fork() falhar, o programa exibe uma mensagem de erro e termina.

Cada filho usa a função execvp() para executar um programa externo, passado como argumento na linha de comando (argv[1], argv[2], argv[3]). Entretanto, o programa interrompe imediatamente cada processo filho após sua criação, enviando o sinal SIGSTOP com kill() para pausá-los.

O programa usa gettimeofday() para obter o tempo atual, e o valor de segundos (tempoAtual.tv_sec) é usado para calcular mod = (tempoAtual.tv_sec) % 60, que representa os segundos do minuto atual. A ideia é sincronizar a execução dos processos com momentos específicos dentro de cada minuto. O programa espera até que o valor de mod seja 0, ou seja, o início de um novo minuto. Quando isso ocorre, o alarme de 120 segundos é ativado com alarm(120) para programar o término da execução após esse tempo. Dentro do loop infinito, o programa verifica o valor de mod em diferentes momentos do minuto para decidir qual processo filho deve ser ativado ou interrompido:

- **mod == 0**: O processo pid2 é pausado e pid3 é iniciado.
- **mod == 25**: O processo pid1 é pausado e pid3 é iniciado.
- **mod == 45**: O processo pid3 é pausado e pid2 é iniciado.
- **mod == 5**: O processo pid3 é pausado e pid1 é iniciado

As variáveis flag1, flag2, e flag3 são usadas para evitar que o mesmo processo seja reiniciado mais de uma vez dentro do mesmo período de tempo. As mensagens de printf são exibidas quando um processo é iniciado, indicando o segundo atual e qual processo está sendo ativado.

O código do processo filho consiste em um laço infinito (while(1)). Dependendo do sinal recebido (SIGSTOP, SIGCONT), o processo filho será pausado ou retomado, conforme o controle do processo pai.