Relatório Trabalho 1

INF1019 - Sistemas de Computação

2017.2

Matheus Rodrigues de Oliveira Leal - 1511316

Felipe Gustavo Pereira Viberti - 1510384

25 de Outubro de 2017

1. Introdução

Esse relatório consiste em explicar e analisar os resultados obtidos para o primeiro de trabalho de Sistemas de Computadores. O programa consiste um interpretador de comandos e um escalonador de programas, ambos programados na linguagem C. O interpretador obtém por meio de stdio os nomes de programas-executáveis e parâmetros, e solicita ao escalonador a execução concorrente dos processos correspondentes a esses programas. O escalonador por sua vez executa de forma intercalada os processos de acordo com uma política de escalonamento que é baseada em 3 filas com níveis de prioridade que dependem do comportamento de cada processo.

2.Arquivos que estão sendo enviados

escalonador.c
escalonador.h
interpretador.c
Fila.c
Fila.h
Processo1.c - Processo Usuário

3.Passos para a execução

Primeiro gerar os .exe para o Processo1.c
$gcc -Wall -o processo1 Processo1.c
Repetir esse passo para a quantidade de processos que se queira escalonador mudando apenas o nome do .exe.
$gcc -Wall -o escalonador escalonador.c Fila.c
$gcc -Wall -o interpretador interpretador.c
Por fim deve-se executar o interpretador(o escalonador será chamado ao final da interação do usuário com o interpretador).
$./interpretador

4. Teste

Para testar o funcionamento do escalonador e interpretador desenvolvidos, foi criado um programa para ser chamado pelo interpretador e simular um processo em execução. Isso só foi possível pois os processos sao escalonado em modo Round-Robin, ou seja, cada processo executando um quantum de tempo. Dessa forma variando o tempo de rajada, você simularia processos diferentes.

a) Dados de entrada

Os dados de entrada foram escolhidos no intuito de observar o escalonador tratando os processos, as interrupções por I/O e as mudança filas.Para o teste cujo log está abaixo foram escalonados 3 processos(processo1, processo2, processo3) cada um com as rajadas de tempo de (3,4,5).

b) Dados de saída

Como podemos ver o escalonador conseguiu gerenciar os processos e as interrupções por I/O

Log de execução do escalonador:

>> Executando processo de nome: processo1
PID:4382
<< Interrompendo processo de nome: processo1 por tempo
Desceu processo para fila 2
>> Executando processo de nome: processo2
PID:4384
<< Interrompendo processo de nome: processo2 por tempo
Desceu processo para fila 2
>> Executando processo de nome: processo3
PID:4386
<< Interrompendo processo de nome: processo3 por tempo
Desceu processo para fila 2
>> Executando processo de nome: processo1
PID:4382

PID:4382
<< Interrompendo processo de nome: processo1 por tempo
ESPERA POR IO
>> Executando processo de nome: processo2
<< Interrompendo processo de nome: processo2 por tempo
PID:4384
>> Executando processo de nome: processo3
<< Interrompendo processo de nome: processo3 por tempo
PID:4386
PID:4384
PID:4386
ESPERA POR IO
ESPERA POR IO
Chegou em terminouIO
>> Executando processo de nome: processo1
<< Interrompendo processo de nome: processo1 por tempo
Desceu processo para fila 2
>> Executando processo de nome: processo1
<< Interrompendo processo de nome: processo1 por tempo
Desceu processo para fila 3
>> Executando processo de nome: processo1
<< Interrompendo processo de nome: processo1 por tempo
PID:4382
PID:4382
PID:4382
PID:4384
Chegou em terminouIO
>> Executando processo de nome: processo2
<< Interrompendo processo de nome: processo2 por tempo
Desceu processo para fila 2
>> Executando processo de nome: processo2
<< Interrompendo processo de nome: processo2 por tempo
Desceu processo para fila 3
>> Executando processo de nome: processo2
<< Interrompendo processo de nome: processo2 por tempo

Chegou em terminouIO
>> Executando processo de nome: processo3
<< Interrompendo processo de nome: processo3 por tempo
Desceu processo para fila 2
>> Executando processo de nome: processo3
<< Interrompendo processo de nome: processo3 por tempo
PID:4386
Desceu processo para fila 3
>> Executando processo de nome: processo3
<< Interrompendo processo de nome: processo3 por tempo

PID:4382
PID:4384
PID:4386
ESPERA POR IO
PID:4384
PID:4386
PID:4384
PID:4386
ESPERA POR IO
ESPERA POR IO
Chegou em terminouIO
>> Executando processo de nome: processo1
<< Interrompendo processo de nome: processo1 por tempo
<< Finalizando processo de nome: processo1
Chegou em terminouIO
>> Executando processo de nome: processo2
<< Interrompendo processo de nome: processo2 por tempo
<< Finalizando processo de nome: processo2
Chegou em terminouIO
>> Executando processo de nome: processo3
<< Interrompendo processo de nome: processo3 por tempo
<< Finalizando processo de nome: processo3
Finalizado

5. Conclusão

No final a execução ocorreu como prevista para o testes realizados, mas vale aqui salientar alguns pontos importantes e interessantes do trabalho. O programa interpretador obtém os comandos por meio de stdio e após guardar os nomes dos programas-executáveis e de seus parâmetros, solicita que seja executado o programa do escalonador, passando para ele um vetor de strings de comandos. Esse por sua vez recebe os comandos como variável de entrada. A função de escalonar em si é um loop que ocorre até que existam processos em uma das 3 filas ou na fila de I/O. Em cada iteração do loop ele verifica de qual fila pegar o processo e se ele ainda não foi finalizado ele dá um SIGCONT no processo.Enquanto ele está executando o escalonador espera pelo tempo do quantum da Fila(ou até que chegue um sinal dizendo que o processo vai entrar em I/O) e depois disso ele checa se o processo entrou ou não em I/O.Se não tiver entrado, ele dá um SIGSTOP. Se tiver, ele insere o processo na fila de processos em I/O e não para o processo. Nesse caso tenho que esperar o processo me dizer se ele de fato entrou em I/O. Como o tempo para cada passagem de sinal estava ocorrendo de forma quase randômica damos um pause para ele esperar receber esse sinal. Reconhecemos que essa não é a melhor forma de fazer isso e que pode acarretar algum erro em determinados testes(já que o pause espera por qualquer sinal e não o sinal específico de SIGUSR1), mas foi a única forma que encontramos para fazer o escalonador operar corretamente. No final de cada iteração do while ele checa para ver se o processo vai descer de fila. Aqui ele não checa se um processo vai subir de fila já que isso só vai ocorrer quando ele retornar de um I/O.Essa checagem, portanto é feita dentro do handler de SIGUSR2 que é o sinal que o processo passa para o escalonador quando ele termina um I/O. Neste handler retiramos um processo da fila de processos em I/O e o inserimos na fila de cima da qual ele estava quando foi para o I/O.Quando todas as rajadas de tempo foram finalizadas e ele retornar do último I/O ele manda o processo filho terminar através do handler de SIGCHLD .Nesse handler temos que verificar se o sinal que foi passado para ele não foi por exemplo SIGSTOP,já que SIGCHLD é sempre chamado quando um processo para ou termina. Se isso não ocorrer e o SIGCHLD for de fato chamado quando o processo acabou damos um SIGKILL nesse processo.