Universidade de Brasília (UnB)
Departamento de Ciência da Computação

Disciplina: Sistemas Operacionais
Prof: Alba Melo
Período: 01/2019

Descrição do Trabalho Prático (01/2019)

Título: Execução Multi-processo postergada

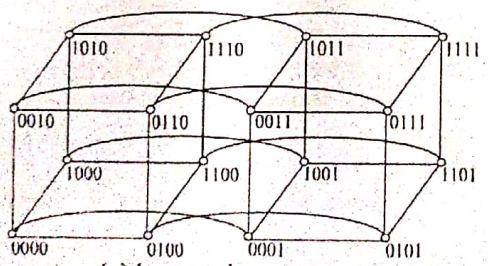
1. Pré-requisitos

O aluno deverá conhecer a linguagem C e chamadas de sistema Unix.

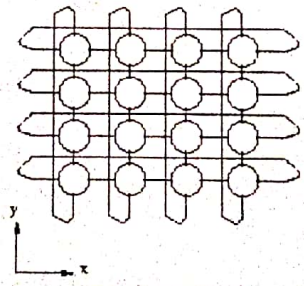
2. Visão Geral :

O presente trabalho consiste da implementação de um gerente de execução de processos múltiplos de maneira postergada. O mecanismo a ser implementado possui função similar ao comando *at* existente no Unix. O sistema terá um processo escalonador multi-processo e 16 (ou 15) nós, cada um com um 1 processo gerente de execução de processos.

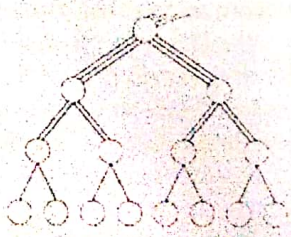
Na inicialização, são criados o processo escalonador e 16 (ou 15) processos gerentes de execução. Os processos gerente de execução estarão conectados segundo uma das topologias abaixo (hypercube, torus ou fat tree), escolhida como parâmetro de execução. O processo gerente de execução 0 estará conectado ao processo escalonador. Os processos conectados podem se comunicar diretamente, ou seja, caso os processos que desejam trocar informação não estejam conectados, eles devem enviar os dados por nodos intermediários conectados. Por exemplo, caso os processos 0 e 5 desejem se comunicar no hipercubo, o dado deve seguir um caminho válido (0 -> 4 -> 5, 0 -> 2 -> 5, etc). Com exceção da fat tree, todos os nós possuem uma conexão entre eles. Na fat tree simplificada, os nodos 0, 1 e 2 possuem duas conexões e os demais nodos possuem uma conexão.



(a) hypercube



(b) torus


(c) fat tree

Via shell, será solicitada a execução postergada de um programa (daqui a *x* segundos) ao processo escalonador. Decorrido o tempo, o processo escalonador solicita que todos os processos gerentes de execução executem o programa em questão, marcando os processos como ocupados. Ao terminar a execução, os processos gerentes de execução informam o processo escalonador do término e são marcados como livres neste momento. Somente os processos livres podem executar processos.

3. Descrição dos Processos :

3.1 Processo de solicitação de execução: Será executado via shell.

a) Sintaxe: **executa_postergado** <seg> <arq_executável>

<seg>: parâmetro obrigatório, a ser fornecido no formato seg (segundos). Especifica o delay de execução em relação à hora corrente.

<arq_executável>: parâmetro obrigatório, nome do arquivo executável a ser executado de maneira postergada

b) Comportamento:

Se não houver parâmetro inválido, o processo executa_postergado atribui um número de job único à tupla <arq_executavel>, <seg> que é guardada.

c) Exemplo:

> **executa_postergado** 5 hello_world
> job=1, arquivo=hello_world, delay=5 segundos

3.2 Processo escalonador de execução de processos: Roda em background.

a) Sintaxe: > escalonador &

b) Comportamento: O processo escalonador é ativado cada vez que for necessário executar um processo com execução postergada. Após *seg* segundos, o processo escalonador de execução solicita que todos os processos gerentes de execução executem o programa <arq_executável>, seguindo as rotas válidas, marcando-os como ocupados. Ao final da execução de um programa, cada processo gerente de execução informa ao processo escalonador a hora início e de término da execução. Nesse momento, o processo gerente de execução é marcado como livre. Ao final da execução de todos os processos, o processo de escalonamento da execução imprime o número do job, a tupla e o turnaround (makespan) da execução.

c) Exemplo:

> job=1, arquivo=hello_world, delay=5 segundos, makespan: 4 segundos

Os processos gerentes de execução executam o programa hello_world às 16:05 e o processo escalonador imprime as informações de controle no final.

Atenção: Podemos ter mais de um processo em espera em um dado instante. Por exemplo, as 16:01 temos a seguinte configuração:

job	arq_exec	dd/mm/yyyy,hh:mm	
5	hello_world	17/04/2019,16:05	(pode ser usado o time_t)
7	teste	17/04/2019,16:02	

Processos Gerentes de Execução:

a) Comportamento: São criados pelo escalonador e possuem duas funções: encaminhar dados entre processos gerentes de execução e executar programas. Para encaminhar dados, o processo gerente deve calcular uma rota que use uma conexão válida. Caso seja solicitada a execução de processos no nodo, o processo gerente de execução, marca o tempo de início e cria um processo filho para execução do programa. Ao término do programa, o processo marca o tempo de fim e envia os tempos de início e fim ao processo escalonador, voltando para esperar novas execuções.

Processo shutdown: Este processo é utilizado para terminar todos os processos. Se houver programas ainda não executados, o processo imprime mensagem informando que os mesmos não serão executados e imprime uma estatística de cada processo que foi efetivamente executado no período de atividade do servidor, contendo pid do processo, nome do arquivo executável, tempo de submissão, tempo de início de execução, tempo de término de execução e makespan.

Os fontes de todos os processos que compõem este gerente devem ser entregues, bem como uma especificação detalhada da implementação: estruturas de dados e mecanismos de comunicação interprocessos utilizados.

IMPORTANTE: A estrutura de dados que representa os processos escalonados para execução postergada **não pode ser escrita em arquivo.** Devem ser utilizados os mecanismos IPC Unix (msg, shm, sem), pipe ou sinais (signal e kill).

BOM TRABALHO !!!!!