Nome n° USP:

ACH 2096 – Laboratório de Sistemas Operacionais Professor: Gisele Craveiro Data: 31/03/2016

**Exercício Programa 1 – Implementação de Simulador Orientado a Eventos**

**Funcionamento do programa**

O sistema é dividido em 7 partes: o simulador propriamente dito; os programas de escalonamento de CPU; o programa de tratamento de dados lidos dos arquivos de entrada; os programas de tratamento de eventos; o programa de memória central; o programa de exibição de mensagens e um arquivo utilitário usado pelos eventos.

**1.Simulador (simulador.c)**

Arquivo principal, que vai executar todas as outras funções do programa.

**2.Escalonamento de CPU**

Um conjunto de três programas, sendo:

* process\_struct.h: esse arquivo contém a struct que representa os processos. Seus atributos são ID, arrive\_time (hora de chegada do processo, para o FIFO), CPU\_burst (quantidade de clocks de CPU que o processo consumira, CPU\_Used (quantidade de clocks de CPU que o processo já consumiu), memory (quantidade de memória alocada para o processo, I\_O (quantidade de requisições I/O), I\_O\_used (quantidade de requisições I/O já feitas), \*processList (lista de ações que o processo deve executar), tempoEspera\_IO (tempo que o processo está na fila de espera por ter feito uma requisição I/O. Caso seu valor seja zero, ele não está esperando e está pronto para ser executado. No RoundRobin, o tempo de espera de todos os processos cujo valor é maior que zero é decrementado. Os processos que estão esperando a mais tempo tem seu tempo de espera zerado mais rapidamente, tornando esse um mecanismo de atendimento de requisições de I/O estilo FIFO) e \*next (ponteiro para o próximo processo. A lista de processos é uma lista ligada)

**3.Tratamento de dados lidos dos arquivos de entrada (entrada.c)**

Basicamente, o programa lê os dados dos arquivos de entrada e converte os dados lidos para informações importantes para o sistema. Ele lê informações úteis para o sistema em geral, como tamanho da memória principal, time slice da CPU para o Round Robin, número de processos etc, além de ler informações dos processos e, após transformar as informações lidas em dados úteis para o programa, organizar os processos de forma adequada.

**4.Tratamento de Eventos**

EV1-NewJobs.c

Responsável por mover novos jobs da fila de entrada para a fila de espera por memória no tempo de entrada do processo.

EV2-MemoryAloc.c

Responsável por alocar memória para um processo, esse programa mantém o programa na espera caso ainda não tenha memória disponível ou move esse processo para a lista de cancelados caso a memória requerida por ele ultrapasse a memória total do sistema.

Poderia-se mover o processo para o fim da fila de espera caso não houvesse memória disponível, mas o simulador está mantendo ele no início da fila para que programas que usem muita memória não precisem esperar até o fim dos outros processos para conseguirem ser alocados.

EV3-CPUExec.c

Programa que gerencia a execução da cpu.

Esse programa consome um dos cpu\_burst do processo, e caso o sistema seja configurado para round-robin, ele move o processo para o fim da fila de programas esperando por cpu caso a quantidade de execuções de cpu desse programa ultrapasse o timeSlice definido pelo roundRobin

EV4-ReleaseCPU.c

Esse programa checa o processo que está no início da fila de execulçao de cpu, e caso esse processo esteja pedindo por IO, ele o move para a fila de IO

EV5-IOExecution.c

Esse programa consome uma IO do primeiro programa da fila de IO.

O IO é feito em estilo FIFO, e mantém programa executando até que ele finalize a quantidade de IO agendada. (Os programas podem ter IO entre as execuções de CPU, nesse caso, o processo fica ‘pulando’ entre as filas de IO e de CPU)

EV6-ReleaseIO.c

Esse programa checa o processo no inicio da fila de IO, caso a próxima ação desse processo for o uso de CPU, o programa move esse processo para a fila de CPU.

e EV7-ReleaseALL.c

Esse programa checa as filas de IO e de CPU e caso algum processo nelas tenha finalizado todas as operações necessárias, ele termina o processo e limpa a memória referente a esse programa.

**5.Memória Central (central\_memory.c)**

Programa responsável pela paginação de memória. Esse programa divide a memória em quadros (frames) de tamanho definido. A alocação de memória é requisitada por páginas, sendo cada página mapeada num frame de memória através da paginação.

**6.Exibição de Mensagens do Sistema (messages.c)**

Programa responsável pela exibição de mensagens sobre o funcionamento do sistema. Ele é capaz de exibir os detalhes de configuração (Tamanho de memória, tamanho de quadros, time slice da CPU, duração da IO e número de processos); as informações de um processo específico (ID, tempo de chegada, CPU Burst, CPU usada, memória, número de requisições I/O, número de rquisições de I/O já feitas); os processos na lista de processos; a lista de frames; as informações de um frame específico (ID, endereço e Job); o mapa de memória; os processos para os quais não há memória disponível; os frames liberados e os processos que forem transferidos de uma fila para outra.

**7.Arquivo Utilitário**

Programa que contém um método que transfere um processo de uma fila de processos para outra. É utilizado nos eventos.

**Participação na Implementação**

Minha parte na implementação do EP foi ...