Universidade de São Paulo (EACH)

ACH2096 - Laboratório de Sistemas Operacionais

Relatório Sistema BASYS

Matheus Preischadt Pinheiro
8623799

supervisionado por
Dr. Gisele da Silva Craveiro

30 de março de 2016

Resumo

Este documento discorre o desenvolvimento de um simulador BASYS descrito no artigo *Computer System Simulation: An Introduction* (M. H. MacDougall), detalhando a estrutura do algoritmo, do código e do software, desde a implementação do sistema em C até a compilação e execução deste utilizando suas devidas entradas.

1 Introdução ao Sistema BASYS

O artigo de MacDougall descreve detalhadamente a criação de um simulador de um sistema operacional em níveis de processamento, memória e disco, a fim de possibilitar o compreendimento do funcionamento dos sistemas operacionais reais utilizados hoje em centenas de dispositivos. MacDougall se utiliza de conceitos básicos do estudo de sistemas operacionais (*clock, paginação, pipelines, etc*) para a criação do simulador BASYS. A ideia gira em torno de uma estrutura que simule um processo (*job*), um processador capaz de interpretar e simular as operações desse processo, uma memória comum de rápido acesso da qual os processos se utilizam para suas operações e uma memória comum de maior capacidade que simula leitura e escrita em disco. Cada operação referente ao processamento desse processo é implementada como um *evento*.

2 Estrutura e Funcionamento

Nesta implementação, o código foi estruturado em três ambientes: principal, estruturas de dados e estruturas do sistema. Cada ambiente é responsável pela caracterização de cada ferramenta do sistema.

O **ambiente principal** é composto apenas do código principal (*main.c*), responsável pelo LOOP do sistema e pela leitura do arquivo de entrada. Como documentado no código, a entrada é interpretada como *input* do teclado e o LOOP principal é responsável por *ouvir e receber* os jobs e eventos do sistemas, bem como passá-los ao simulador para tratamento. O **ambiente estruturas de dados** é composto pelos três arquivos que implementam a fila utilizada no sistema. Por fim, o **ambiente de estruturas do sistema** é composto por um *header* que descreve as estruturas do escalonador, dos jobs e dos eventos e um arquivo que implementa o comportamento do escalonador no simulador.

2.1 Fila

Como o sistema BASYS se utiliza de quatro filas para simular o comportamento do processador, memória, disco e escalonador de um sistema operacional, sendo que três dessas são filas de processos e uma é uma fila de eventos, foi utilizada uma fila genérica, implementada para um trabalho de *Organização e Arquitetura de Computadores*. A implementação é composta por três arquivos:

- *filaPub.h:* descreve a estrutura para criação da fila genérica, os protótipos das funções *públicas*, serve como uma forma simples de encapsulamento das manutenção e estrutura da fila.

- *filaPriv.h:* descreve a estrutura e funcionamento da fila, as referências para cabeça (*frente*, no código), cauda e tamanho da fila, bem como seus nós genéricos, contendo uma referência aos nós próximo e anterior, bem como os dados genéricos a serem armazenados. Este *header* é conhecido apenas pelo arquivo que implementa a fila, permitindo, como supracitado, um encapsulamento simples da fila.

- *fila.c:* implementa a série de funções para utilização da fila genérica. Foi adicionado para este software uma função de inserção em prioridade, utilizada na fila de eventos (pois estes são inseridos baseados no tempo em que deverão rodar). O algoritmo implementado se utiliza de uma função de *callback* utilizada para comparar os valores que serão utilizados como prioridade, inserindo dessa forma. Foi adicionada, também, uma função de impressão que, da mesma forma, por ser genérica, se utiliza de uma função de *callback* que descreve a forma de impressão e o tipo do dado a ser impresso.

2.2 Escalonador, Job e Event

MacDougall deixa bastante claro o que é necessário para cada estrutura básica do sistema. Todas as características do escalonador, dos *jobs* e dos *eventos* são apresentadas no corpo do artigo (identificação, tempo de cpu, memória utilizada, etc).

- *structs.h* descreve as estruturas dos *jobs* e *eventos* de acordo com as características citadas no artigo, implementando todas as variáveis necessárias para a entrada e simulação no sistema. Descreve, também, a estrutura do escalonador, contendo as quatro filas (CPU, memória, disco e eventos), além as flags de utilização da CPU e disco, além dos protópicos das funções *públicas* do escalonador.

- *scheduler.c* implementa as funções de tratamento dos oito eventos. O *scheduler* (escalonador) é, também, responsável pela criação e manutenção das filas. A implementação foi feita da seguinte maneira: cada evento está ligado a uma função do arquivo (o evento 3, por exemplo, é tratado pela função *eventCpuRequest*), que é chamada no LOOP principal. A função trata o evento de acordo com as instruções do artigo, utilizando-se da função *scheduleEvent* para agendar futuros eventos para o *job* sendo processado ou, no caso do evento 7 (finalização do job), encerrando-o. As funções *compareEvent*, *printEvent* e *eventRrInterruption* foram implementadas, respectivamente, como função de *callback* para comparar os tempos agendados para a fila de prioridade, função de *callback* descrevendo o tipo

de dado a ser impresso pela função *imprimeFila* da lista genérica e tratamento da troca de processos no *Round-Robin* pedido no enunciado do trabalho.

2.3 Informações Adicionais

O funcionamento do sistema está explicado com mais detalhes na forma de comentários dentro dos arquivos fontes e nos commits contidos no repositório mp-pinheiro/EP1-BASYS, a documentação discorre o funcionamento de cada evento, ocupação e liberação de CPU e disco quando necessário, alocação e liberação de memória e afins. No corpo do código, foram especificados locais onde podem ser implementados tempos de delay para simular situações mais próximas da realidade (como dificuldades de busca em memória, por exemplo). Também foram seguidas as instruções do artigo na implementação do *listener* de eventos (além da sugestão da professora Dra. Gisele da Silva Craveiro de utilizar um *switch case*), permitindo a fácil adesão de novas interrupções e eventos no sistema.

Referências

[1] M. H. MacDougall. *Computer System Simulation: An Introduction.* Journal ACM Computing Surveys (CSUR) Surveys Homepage archive Volume 2 Issue 3, Sept. 1970, Pages 191-209