wxEscal$_{Proc}$ — um simulador de políticas de escalonamento multiplataforma

Adriano Arlei de Carvalho
Anderson de Rezende Rocha
Andréia Scheneider
Antonio Galvão de Rezende
Júlio César Alves
Curso de Ciência da Computação*
{arlei,undersun,andreia,jcalves,galvao}@comp.ufla.br

18 de junho de 2003

Resumo

A existência de uma ferramenta gráfica como auxílio para se aprender determinados assuntos pode facilitar muito o aprendizado do aluno. Criou-se, através deste trabalho, uma ferramenta gráfica para o aprendizado de políticas de escalonamento, com o intuito de auxiliar o professor na hora da ministração desse assunto.

1 Introdução

Escalonamento de CPU é a base dos sistemas operacionais multiprogramados. Através da troca de uso da CPU entre os processos é possível fazer com que o computador fique mais eficiente. O maior problema é decidir qual processo deve ser executado primeiro. Há diferentes algoritmos que tratam disto, dentre eles estão o *First-Come-First-Served*, *shortest-job-first*, *Roudin Robin* e *shortest-job-first* preemptivo.

1.1 First Come First Served

É o mais simples de todos os algoritmos de escalonamento. Ele parte da idéia da fila (primeiro a entrar é o primeiro a sair). Quando um processo é criado ele vai para a fila de espera. Quando a CPU estiver livre, o processo que a ocupará será o primeiro da fila. O próximo processo só sairá da fila e ocupará a CPU quando o primeiro tiver sido executado totalmente.

*Universidade Federal de Lavras, MG. Disciplina de Sistemas Operacionais lecionada pelo prof. Ricardo Martins de Abreu Silva

1.2 Round Robin

Round-Robin é um dos mais antigos e simples algoritmos de escalonamento. É largamento usado, e foi projetado especialmente para sistemas *time-sharing*. A idéia do algoritmo é a seguinte. Uma pequena unidade de tempo, denominada quantum, é definida. Todos os processos são armazenados em uma fila. O escalonador da CPU percorre a fila, alocando a CPU para cada processo durante um quantum. Mais precisamente, o escalonador retira o primeiro processo da fila e procede à sua execução. Se o processo não termina após um quantum, ocorre uma preempção, e o processo é inserido no fim da fila. Se o processo termina antes de um quantum, a CPU é liberada para a execução de novos processos. Em ambos os casos, após a liberação da CPU, um novo processo é escolhido na fila. Novos processos são inseridos no fim da fila.

1.3 Shortest Job First

A idéia deste algoritmo é executar sempre o processo que tenha menor tempo de execução. Desta forma, todos os processos que vão sendo criados são colocados em uma lista. Sempre que a CPU esta livre o menor processo existente na lista tomará conta da mesma.

1.4 Remaining Job First

A diferença entre este algoritmo e o anterior é que este não executa o processo de uma vez. A cada intervalo de tempo a lista é consultada e se existir um processo cujo tempo necessário para ele ser executado menor que o tempo restante para o processo que está ocupando a CPU ser executado, este processo de menor tempo ocupará a CPU, enquanto que o outro retornará a lista e esperará por sua vez.

2 Visão geral

O simulador wxEscal$_{Proc}$ foi desenvolvido no âmbito de permitir aos alunos da disciplina de Sistemas Operacionais acompanharem visualmente a simulação de processos fictícios no computador. Para cada conjunto de processos o aluno pode visualizar como seria o resultado deste conjunto se este fosse submetido a uma das políticas de escalonamento implementadas.

Foram implementadas 4 políticas. São elas:

1. FCFS (*First Come First Served*) ou simplesmente Fila
2. Round Robin com *quantum* variável
3. SJF (*Shortest Job First*)
4. RJF (*Remaining Job First*)

3 Compilação do Programa

O programa desenvolvido é multi-plataforma; pode ser compilado em Linux e Windows. Para que isso fosse possível, foi utilizada linguagem de programação C++, a biblioteca STL e a biblioteca gráfica wxWindows (`http://www.wxwindows.org`).

Para compilar o wxEscalProc em Linux digite na linha de comando:

```
make -f makefile.unx
```

Em Windows digite:

```
make -f makefile.win
```

4 O funcionamento

O programa tem as seguintes opções de menu: Configurar Processos, Escolher Política, Tic, Sobre e Sair.

Configurar Processos

Permite a inserção dos processos a serem escalonados. O usuário deve digitar o tempo de criação e de processamento de cada processo. Se quiser adicionar mais um processo basta clicar no botão Adicionar. Quando terminar deverá clicar no botão Terminado. Será então apresentada a janela para a escolha da política de escalonamento.

Escolher Política

Permite a escolha da política de escalonamento. O usuário clicará na política desejada e escolherá entre os modos de simulação passo-a-passo e automático. No modo passo-a-passo cada vez que ele clicar no menu Tic (ou pressionar a tecla de atalho $F7$) será executada uma unidade de tempo da simulação. Já no modo automático a simulação é executada diretamente até o final.

Tic

Executa uma unidade de tempo da simulação. Caso a simulação já tenha terminado é apresentada uma tabela com os tempos de criação, processamento, finalização e turnaround de cada processo.

Sobre

Exibe informações sobre o desenvolvimento do programa.

5 A implementação

5.1 As políticas

Todas as políticas implementadas mantêm uma referência a uma lista de processos criada pelo usuário. Esta referência funciona como se fosse o repositório de processos. Ao final da execução basta o aplicativo utilizando as políticas recuperar esta lista de processos modificada por uma política em particular.

Toda classe representando uma política possui um *timeline* que é o tempo atual de todos os processos. Isto quer dizer que se há 3 processos e a soma dos tempos de processamento destes processos é 14, logo *timeline* deve variar de 1 a 14. Além disso, as classes mantém uma fila de processos que é a estrutura de dados que representa a prioridade de execução dos processos desta política de escalonamento. A cada *t* unidade de tempo pode-se executar um *tic*. A cada *tic* a política é responsável pela atualização das devidas variáveis envolvidas bem como por realizar ou não uma preempção.

5.2 As classes

Para implementar as quatro políticas criou-se as seguintes classes:

1. clProcesso
2. clFcfs
3. clSJF
4. clRJF
5. clRoundRobin

5.2.1 clProcesso

É a representação do processo fictício. Isto é, enquanto um objeto desta classe existir o processo ainda está sendo executado no computador.

Cada processo é representado por:

- um identificador único que lhe é atribuído no ato de sua criação pelo simulador;
- um tempo de criação que é fornecido pelo usuário;
- um tempo de finalização também fornecido pelo usuário;
- um *turn-around* calculado pelo simulador de acordo com o andamento de cada política simulada;
- um tempo de finalização também calculado pelo simulador
- um status que guarda o estado corrente do processo para que este possa ser devidamente preemptado e, quando chamado novamente à execução, possa recuperar sua consistência;
- um identificador que mostra se o processo é o processo atualmente usando o processador.

5.2.2 clFcfs

A decisão se um processo deve ou não passar a ser ativo é dada pela própria fila utilizada pela política. A cada *tic* percorre-se a lista de processos recebida por referência e verifica-se se há processos sendo criados naquele momento. Caso positivo, os processos são inseridos na fila de prioridades da política. Caso o processo atualmente no processador termine automaticamente a lista de processos é atualizada e o novo processo a "ocupar" o processador será o cabeça da fila de prioridades.

5.2.3 clSJF

Esta política é bastante semelhante à política de fila descrita no item 5.2.2. No entanto, assim que um processo que esteja "ocupando" o processador acaba o próximo processo a assumir o processador será o processo com o menor tempo de processamento atualmente na fila de prioridades. Deste modo, a cada término de um processo basta fazer uma busca na fila de prioridades em busca do processo com este menor tempo.

5.2.4 clRJF

A decisão se um processo deve ou não passar a ser ativo é dada pelo processo com menor tempo de execução presente na fila de prioridades. Deste modo, a cada *tic* busca-se na lista de processos todos aqueles que estiverem sendo criados naquele momento e estes são inseridos na fila de prioridades. Em seguida, faz-se uma busca na fila de processos e escolhe-se aquele que tenha o menor tempo de processamento e faz-se a preempção.

5.2.5 clRoundRobin

A fila de prioridades funciona como uma fila circular. Deste modo, executa-se um *tic* para o processo atualmente no processador e este é colocado no final da fila de prioridades. O próximo processo a "ocupar" o processador será aquele que estiver na cabeça da fila de prioridades. Em seguida, busca-se todos os processos sendo criados naquele momento a partir da lista de processos e estes são inseridos na fila de prioridades.

Para a simulação com *Round Robin* de *quantum* variável a diferença é que ao invés de permitir a possibilidade de preempção a cada *tic* esta só pode ocorrer ao término de cada *quantum*.

O resultado da implementação pode ser visto na figura 1

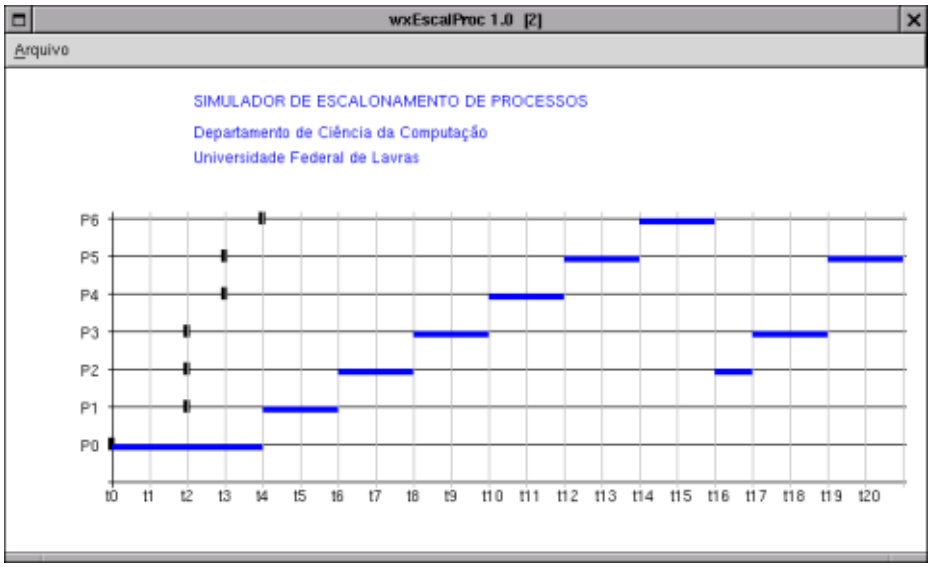

Figura 1: Exemplo de uma tela do wxEscal$_{Proc}$

6 Considerações finais

Os autores deste trabalho acreditam que esta ferramenta poderá contribuir de maneira significativa para o aprendizado de processos de escalonamento. Porém, isto

ainda não pode afirmado, pois não há um estudo de caso realizado. Sugere-se como prosseguimento deste trabalho um estudo de caso para afirmar a tese de que a wxEscal$_{Proc}$ realmente é um auxiliador no processo de aprendizado.

Referências

[DEITEL & DEITEL] DEITEL, H. M. & DEITEL, P. J., *C++ Como Programar*. Bookman, 2001.

[JOSUTTIS] JOSUTTIS, N. M., *The C++ Stardard Library: A Tutorial and Reference*. Addison Wesley, 2000.

[SILBERSCHATZ] SILBERSCHATZ, A., *Operating system concepts*, 5. ed. Reading, Mass: Addison-Wesley, 1998.