Escalonador por processo de loteria

Rodrigo Ediel Vogt[1]

[1]Ciência da Computação - Universidade Federal da Fronteira Sul (UFFS)
Chapecó – Santa Catarina – Brasil

rodrigo.vgt@gmail.com

Abstract. *This article describe a schedule implemented through lottery model, in which the choice of processes occurs through the random selection of tickets, implemented on xv6[xv6 MIT] operational system base.*

Resumo. *Este artigo descreve um escalonador implementado atraves do modelo de loteria, no qual a escolha dos processos ocorre através da seleção randomica de bilhetes, implementado no sistema operacional xv6[xv6 MIT] .*

1. Introdução

O sistema operacional xv6[xv6 MIT] é um sistema educacional, desenvolvido pelo MIT(Massachusetts Institute of Technology). Por ser um sistema baseado em código aberto, é possivel realizar modificações em seu código fonte, fundamental para o aprendizado do funcionamento dos aspectos mais importantes de um sistema operacional. Perante este sistema operacional, incumbiu-se a tarefa de realizar a implementação de um escalonador que agisse pelo sistema de loeteria, com base em quantidade de bilhetes.

2. Escalonador

O escalonador, parte do sistema operacional responsavel por qual será o proximo processo a executar[Teixeira], que age por meio de um algoritmo de escalonamento. Este algoritmo de escalonamento é o enfoque deste trabalho, que tem por objetivo final implementar um escalonar pelo método de loteria.

3. Bilhetes

O sistema de loteria ”é uma modalidade popular de jogo de azar que consiste no sorteio aleatório de algo pré-estabelecido, normalmente um número, em troca de um prêmio”[Dicionario Português], sendo isso valido também para o escalonador por loteria, no qual o prêmio é acesso ao processador.

Pelo método de bilhetes, é possivel implementar também um método de prioridade, já que o processo que tiver mais bilhetes tem mais chances de ser selecionado, e portanto, tem mais chance de permanecer mais tempo.

O código utilizado para realizar o sorteio, foi encontrado no próprio xv6, no arquivo usertests.c, e sofreu uma leve modificação, para que o valor aleatorio gerado nao extrapolasse os valores referentes ao numero máximo de tickets.

```
unsigned long randstate = 1;
unsigned int
rand(int total)
{
  randstate = (randstate * 1664525 + 1013904223)%total;
  return randstate;
}
```

Nesta função, durante a primeira utilização, o valor da variável randstate será 1, porém, durante as próximas utilizações, devido à multiplicação por um numero gigante, junto à adição de um numero primo, e também à divisão pelo limite debilhetes realizada, o numero tende a não se repetir seguidamente,

4. Fork

”Fork é uma operação onde um processo cria uma copia dele mesmo.”[Wikipedia]. O processo de fork original do xv6, chamava uma função void, ou seja, não passava nenhum atributo. para a implementação do escalonador por loteria, foi necessario mudar a estrutura do fork utilizado pelo sistema para comportar a inclusão do dado referente ao numero de bilhetes.

5. Funcionamento do escalonador

A base de funcionamento do escalonador é bem simples. Assim que é chamado, o escalonador percorre o vetor de processos procurando por todos os processos que estejam no

estado RUNNABLE, o que significa que o processo ja esta pronto para ser executado. A seguir, se o processo estiver pronto, o escalonador verifica o numero de bilhetes que este processo possui, e adiciona esse valor na variável total, que conta a quantidade total de bilhetes.

```
int totaltickets(void) {
struct proc *p;
int total = 0;
for (p = ptable.proc; p < &ptable.proc[NPROC]; p++) {
if (p->state == RUNNABLE) {
total += p->tickets;
}
}
cprintf("Total de tickets aptos a participar do sorteio: %d ", total);
return total;
}
```

Após a realização da contagem de total de bilhetes, é feito o sorteio através da função rand, já citada acima, a qual retorna o bilhete vencedor. Assim que o vencedor é decidido, o escalonador volta para a lista de processos hábeis, procurando pelo vencedor. De forma inversa a qual é feita a contagem de bilhetes e feita a checagem do vencedor. O escalonador percorre toda a lista novamente, descontando o valor dos bilhetes dos processos, e assim que o valor atinge 0, é encontrado o processo detentor do bilhete premiado, e ele prossegue para a execução.

```
qttickets = totaltickets();
if (qttickets > 0) {
sorteado = rand(qttickets);
if (sorteado < 0) {
sorteado = sorteado * -1;
cprintf("rand negativo\n");
}
if (qttickets < sorteado) {
sorteado = sorteado % qttickets;
}

for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
if (p->state == RUNNABLE) {
sorteado = sorteado - p->tickets;
}
if(p->state != RUNNABLE || sorteado >= 0) {
continue;
}
proc = p;
switchuvm(p);
p->state = RUNNING;
swtch(&cpu->scheduler, p->context);
switchkvm();
```

6. Conclusão

O projeto do escalonador ocorreu com completo sucesso. Conforme o esperado os precessos detentores de maior numero de bilhetes terminam sua execução quase sempre primeiro. A implementação do trabalho foi de suma importancia para o compreendimento de funções de execuções do sistema, como a função fork, a qual foi a função fundamental na realização da criação do arquivo de teste para o funcionamento do projeto.

Referências

Dicionario Português. Loteria [on-line] definição. Disponível em: http://dicionarioportugues.org/pt/loteria. Accessed: 2017-05-21.

Teixeira, M. A. Escalonador de procesos. Disponível em: http://ctd.ifsp.edu.br/ marcio.andrey/images/Escalonamento-Processos-IFSP.pdf. Accessed: 2017-05-21.

Wikipedia. Fork(system call). Disponível em: https://en.wikipedia.org/wiki/Fork(systemcall). Accessed: 2017-05-21.

xv6 MIT. xv6 operational system. Disponivel em: git://github.com/mit-pdos/xv6-public.git. Accessed: 2017-05-21.