Sistemas Operacionais
Trabalho Prático 2

Luís Felipe Ramos Ferreira

lframos_ferreira@outlook.com

1 Introdução

O Trabalho Prático 2 da disciplina de Sistemas operacionais teve como proposta o estudo e modificação dos algoritmos de escalonamento de processos presente no *kernel* do sistema operacional XV6.

O repositório onde está armazenado o código utilizado durante o desenvolvimento desse projeto pode ser encontrado neste endereço.

2 Respostas

1. Qual a política de escalonamento é utilizada atualmente no XV6?

   A política de escalonamento utilizada no XV6 é uma política de

   Round Robin sem prioridades, ou seja, o escalonador irá checar continuamente a lista de processos disponíveis para serem executados e irá fornecer um tempo de processamento a cada um deles, sem que exista uma política de priorização dentre um processo a outro. O algoritmo é simples de se implementar, simples de compreender e não causa inanição aos processos, embora possua pontos negativos como os gargalos causados pela constante troca de contexto, a depender do tempo que cada processo terá para execução na CPU. É um algoritmo preemptivo, uma vez que força a saída de um processo da CPU caso o limite de tempo tenha sido atingido.

2. Quais processos essa política seleciona para rodar?

   A política citada seleciona os processos que estão disponíveis para serem executados conforme eles são checados na lista de processos disponíveis. Não há um tipo de prioridade estabelecida em cima sobre os processos, ou seja, os processos terão uma certa quantidade de tempo a cada momento que o escalonador encontrá-los na lista de processos disponíveis. É importante frisar que o escalonador irá checar `apenas` os processos marcados como disponíveis para serem executados, ou seja, processos dormindo ou esperando algum I/O não receberão tempo de processamento da CPU a menos que estejam prontos para serem executados e marcados como tal na lista de processos do sistema.

3. O que acontece quando um processo retorna de uma tarefa de I/O?

   O processo é marcado como *RUNNABLE*, isto é, está pronto para executar e entra para a lista de processos que podem ser executados. Assim, ele eventualmente será escolhido pelo escalonador para começar a rodar.

4. O que acontece quando um processo é criado e quando ou quão frequente o escalonamento acontece?

   Quando um processo é criado, uma referência para ele é criada no espaço de memória do sistema operacional, e a esse processo deve ser alocado um espaço de memória de usuário onde irá estar armazenado seu identificador, código, dados, pilha de execução e *heap*. Um processo pode ser criado no XV6 por meio da chamada de sistema *fork()*, que irá criar uma cópia do processo que fez a chamada da função. Para executar um novo programa, a chamada de sistema *exec()* deve ser utilizada.

O processo de escalonamento acontece, na implementação original do XV6, a cada 1 tick do clock. No entanto, esse parâmetro pode ser modificado para alterar o período entre as preempções realizadas pelo escalonador.

3 Algoritmos implementados

3.1 Escalonador

Uma das requisições do trabalho foi a de implementar uma modificação na política de escalonamento do *kernel* do XV6. Em particular, foi proposta a implementação de um escalonamento por meio de filas multinível. Para isso, a função *scheduler()* do arquivo *proc.c* foi alterada de modo a satisfazer essa política. Em suma, a nova implementação agora irá checar toda a lista de processos em busca daquele processo que possui a maior prioridade e, ao fim da busca, esse processo será o próximo a ser executado. Não foi utilizada nenhuma estrutura de dados customizada para facilitar essa checagem, como uma fila de prioridades. A busca pelo processo de maior prioridade é feita de forma linear na lista de processos, para facilitar a implementação.

3.2 Chamadas de sistema

Ao todo, quatro novas chamdas de sistema foram implementadas para facilitar o desenvolvimento do trabalho, e o propósito de cada uma delas está desrito a seguir.

3.2.1 change_prio()

Conforme especificado no enunciado, a chamada de sistema *change_prio()* deve ter utilizada para mudar a prioridade do processo atual. Sua implementação é simples, e basta trafegar por toda a lista de processos, encontrar o proesso com o identificador correto e mudarsua prioridade conforme o parâmetro desejado.

3.2.2 wait2()

Conforme também especificado no enunciado, a chamada de sistema *wait2()* deve ser utilizada como uma extensão da chamada de sistema *wait()* mas com algumas ações a mais. Em particular, ela deve também atribuir a três posições de memória os valores totais de tempo que o processo passou nos estados *READY*, *RUNNING* e *SLEEPING*, de modo que tais valores possam ser utilizados posteriormente na análise de dados das modificações propostas para o escalonador.

3.2.3 yield2()

A chamada de sustema *yield2()* se trata apenas de uma solução paliativa para que a função *yield()* seja utilizada nos programas do tipo S-CPU. Sem defini-lá, não era possível utiliar *yield()* em um programa de usuário pois não existia uma interface definida para seu uso em espaço de usuário no *kernel* do XV6.

Em suma, a chamada *yield2()* apenas chama a função *yield()*.

3.2.4 set_prio()

A chamada de sistema *set_prio()* deve ser utilizada para modificar a prioridade de um processo

4 Análise de resultados

A p

5 Conclusão

Em s

6 Referências

- Livros:
  - Tanenbaum, A. S. & Bos, H. (2014), Modern Operating Systems, Pearson, Boston, MA.
  - Abraham Silberschatz, Peter Baer Galvin, Greg Gagne: Operating System Concepts, 10th Edition. Wiley 2018, ISBN 978-1-118-06333-0
  - Arpaci-Dusseau, Remzi H., Arpaci-Dusseau, Andrea C.. (2014). Operating systems: three easy pieces.: Arpaci-Dusseau Books.
- Web:
  - *xv6: a simple, Unix-like teaching operating system*
- Youtube:
  - Jacob Sorber
  - Code Vault
  - hhp3 xv6 *kernel playlist*