Universidade de São Paulo
Instituto de Matemática e Estatistica
IME

EP SO - Escalonamento de processos

Patrícia da Silva Rodrigues (nºUSP 11315590),
Gabriel Ferreira de Souza Araujo (nºUSP 12718100)

A
2023

1 Parte I

No diretório **/usr/src/servers/is/dmp_kernel.c** Editamos a função privileges_dmp e usamos o exemplo do que ocorria na função sched_dmp e atráves dela corremos os processos e printamos a fila de processos por ordem de prioridade.

2 Parte II

- Ir no diretorio /usr/src/servers/pm/table.c

  nesse arquivo foi incluido os handlers para os numero 69

- no /usr/src/include/callnr.h a gente definiu a chamada LOCKPRIORITY como 69

- no /usr/src/servers/pm/proto.h criamos o prototipo do tipo _PROTOTYPE(int do_lockpriority, (void)))

- /usr/src/servers/pm/misc.c foi includo o codigo do lockpriority

- /usr/src/include criamos o arquivo lockprioritylib.h. Esse arquivo tem a syscall para LOCKPRIORITY

- modificamos a parta /usr/src/kernel modificamos alguns arquivos para adicionar a sys_lockpriority, mais expecificamente system.c, system.h e config.h e criamos um novo arquivo do_lockpriority dentro da pasta /usr/src/kernel/system;

- modificamos a /usr/src/lib/syslib/criamos um arquivo chamado sys_lockpriority.c

- adicionamos o prototipo da sys_lockpriority no arquivo /usr/src/include/minix/syslib.h

- criamos um arquivo teste dentro da /usr/src/include/teste.c nesse arquivo basta dar um ./teste. Para trocar a prioridade, basta modificar o teste.c nesse diretório.

3 PARTE 3

Para modificação da política de escalonamento, foi modificada a função sched() no arquivo usr/src/kernel/proc.c. Esse código é parte de um sistema operacional, mais especificamente do escalonador de processos. Ele é responsável por decidir qual processo será executado a seguir, baseado em um conjunto de critérios, como a prioridade do processo, o tempo que ele já utilizou do seu quantum (período de tempo que o processo tem para executar antes de ser interrompido) e a presença de outros processos em filas de prioridade mais alta. A função sched recebe três argumentos: um ponteiro para a estrutura de processo do processo atual (rp), e dois ponteiros inteiros para as variáveis queue e front. A variável queue é usada para indicar em qual fila de prioridade o processo atual deve ser colocado, e front é um indicador se o processo deve ser adicionado no início ou no final dessa fila. O código começa verificando se o processo atual já utilizou todo o seu quantum (ou seja, se o tempo restante para o processo executar é zero). Se sim, ele reseta o contador de tempo restante para o

valor do quantum padrão para esse processo (rp-¿p_quantum_size), e verifica se o processo está em uma das filas de prioridade de 7 a 14. Essas filas são usadas para processos interativos, como o shell, que precisam de uma resposta rápida do sistema operacional. Se o processo está em uma dessas filas de prioridade, o código percorre a lista de processos procurando o próximo processo na próxima fila de prioridade. Ele ignora o próprio processo atual e verifica se o próximo processo encontrado está na próxima fila de prioridade (que é a atual mais 1), e se está dentro do intervalo de filas de prioridade de 7 a 14. Se ele encontra um processo nessa condição, ele indica que o próximo processo a ser executado é o encontrado, setando queue para a prioridade do próximo processo e front para 1 (indicando que o processo deve ser adicionado no início da fila). Se nenhum processo é encontrado na próxima fila, ele continua com o processo atual, adicionando-o no final da mesma fila. Se o processo atual ainda tem tempo restante no seu quantum, o código simplesmente indica que ele deve continuar na mesma fila, adicionando-o no final da fila com a prioridade atual. Abaixo temos o código modificado da função sched():