Relatório EP2- MAC0422

Thiago Santos Teixeira - nUSP 10736987

Wander Souza - nUSP 10737011

Tarefa 1:

Para realizar a primeira tarefa do EP, precisávamos criar uma nova fila de prioridade para o MINIX. Pensando nisso, começamos a editar o arquivo header `proc.h`, já que segundo o livro as filas de processos eram definidas seguindo o padrão da seguinte imagem:

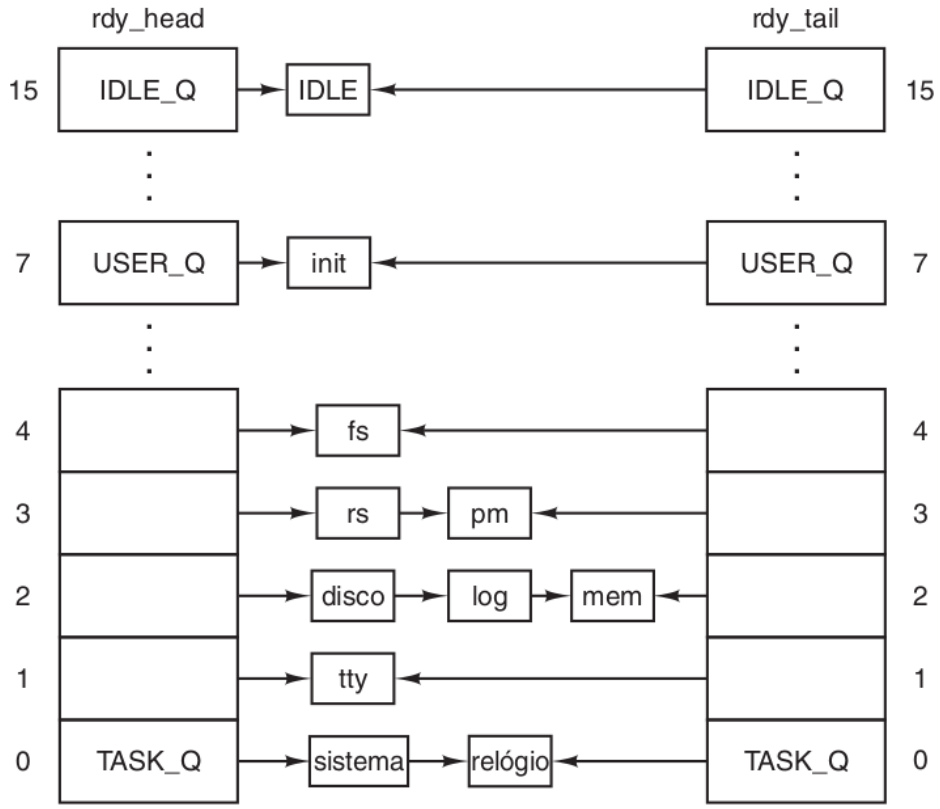

Figura 2-43 O escalonador mantém 16 filas, uma por nível de prioridade. Aqui está mostrado o enfileiramento inicial de processos quando o MINIX 3 é inicializado.

A partir dela, começamos a identificar no código os parâmetros que teríamos que alterar para que nossa nova fila tivesse prioridade entre os processos de usuário e o IDLE, que deve ter sempre prioridade mínima (ou seja, a de maior número). Primeiro, alteramos o parâmetro `NR_SCHED_QUEUES` , que recebe a propriedade mínima + 1, ou seja, a quantidade de filas que o sistema terá, e como iremos inserir um novo nível de prioridade, teremos que o incrementar em uma unidade, indo de 16 para 17.

O mesmo foi feito com `IDLE_Q` , que representa o nível do IDLE, que deverá sempre ser o mínimo e então também precisa ser incrementado, mas dessa vez de 15 para 16. Após isso, para definir a nova fila, apenas inserimos a linha `#define BATCH_Q 15`, assim criando uma nova fila com prioridade 15, entre a `MIN_USER_Q`, que é 14 e a do IDLE, que é 16.

####

Tarefa 2:

Para realizarmos a segunda tarefa do EP, nos baseamos inicialmente, no tutorial publicado pelo monitor no forum do PACA. Assim, iniciamos criando um handler para nossa nova chamada, porém diferente do tutorial já que nele, o handler é implementado no `FileSystem`, enquanto no nosso SO implementamos no `ProcessManager`, mais precisamente em `/usr/src/servers/pm/table.c` editando a entrada 69, que estava inutilizada e adicionando nosso novo protótipo `do_batch`.

O segundo passo foi adicionar o protótipo no arquivo header `proto.h` encontrado no mesmo diretório. Para isso, simplesmente adicionamos a linha `_PROTOTYPE( void do_batch, (void));` embaixo da sessão reservada para o arquivo `misc.c`, que escolhemos para implementar o corpo de nossa função para evitar ter de realizar mudanças no `MAKEFILE`. Construímos a linha seguindo o exemplo do tutorial, ou seja, `data_type nome_da_funcao, (entrada_da_funcao))`.

Já no arquivo `misc.c`, implementamos a nossa função `do_batch()`. Nela, chamamos a `proc_from_pid()` que receberá da mensagem* de sistema o PID do processo que desejamos escalonar, e devolverá seu índice/posição na tabela de processos. Com ela, usamos `mp.parent` para encontrar o PID de seu processo pai, e assim, através da função `getpid()`, verificamos se o processo atual é o pai através de seu PID, satisfazendo assim a condição do enunciado que diz que a nossa chamada só pode ser executada pelo processo pai. Caso a condição seja satisfeita, realizamos a nossa chamada `sys_batch()` criada mais adiante, que recebe o índice devolvido em `proc_from_pid()`.

***Mensagem declarada em `/src/lib/posix/_batch.c`.**

Continuando com o tutorial, adicionamos `#define BATCH 69` em `/src/include/minix/callnr.h` e incrementamos o valor de `NCALLS`, e então partimos para criar a nossa kernel call, que será responsável pelo escalonamento em si.

Os primeiros passos foram a criação do arquivo `sys_batch.c`, que será um wrapper para a nossa kernel call, adicioná-la em `/src/include/minix/com.h`, com valor 31, além de incrementar o valor de `NR_SYS_CALLS`. Depois criamos um handler em `/usr/kernel/system.h` e partimos para montar o código da chamada em `/usr/kernel/system/do_batch.c`, nele, nos baseamos na função enqueue() em `proc.c`, ela usa o índice do processo na tabela para jogá-lo no final da fila de prioridade 15, que é a nossa BATCH.

Depois mapeamos o request type em `kernel/system.c` e adicionamos na system library em `usr/lib/syslib/sys_batch.c`. Além de um protótipo em `/src/include/unistd.h` e editamos os devidos Makefiles, porém, não conseguimos demarcar com comentários as linhas que adicionamos, pois estavam causando problemas na hora de compilar os arquivos.

Problemas encontrados:

Porém mesmo após fazer tudo isso, nosso minix ainda apresentava problemas na hora de realizar a system call, ao compilarmos e rodarmos o SO, ele não reconhecia a call batch(pid), apenas a do_batch(pid), que ao ser executada, por alguma razão nada acontecia e o Minix travava por completo. Passamos horas tentando resolver o que estava errado, mas ainda assim não conseguimos identificar a raiz do problema.

Estando nessa situação complicada, acabamos deixando de implementar a unbatch(pid), já que a lógica de sua implementação seria muito similar, quase idêntica à da tarefa 3, e portanto, tentar criá-la sem ter conseguido a sua irmã seria um desperdício de tempo e esforço, já que apenas criaria problemas adicionais. Porém, ainda decidimos enviar o EP, já que acreditamos que nosso esforço possa valer alguma

coisa, além de que precisamos de feedback para que possamos realizar os devidos ajustes e podermos fazer os próximos EPs de maneira mais adequada.