EP2 – MAC422 - Sistemas Operacionais

Daniela Gonzalez Favero 10277443
Felipe Castro de Noronha 10737032

As quatro tarefas pedidas pelo enunciado foram implementadas assim:

1. A macro

Acrescentamos uma nova fila de prioridade ao Minix com a macro `BATCH_Q` em *proc.h*, para isso, diminuimos o valor que a prioridade máxima do usuário pode assumir, com isso, `BATCH_Q` assume o valor 14.

2/3. As chamadas de sistema

Definimos as macros das chamadas de sistemas no arquivo `include/minix/callnr.h` e prototipamos as funções em `servers/pm/proto.h`. Também alteramos o vetor de chamadas de sistemas em `servers/pm/table.c`.

Os arquivos `_batch.c` e `_unbatch.c` em */usr/src/lib/posix/* são os envolucros referentes a chamada de sistema, de modo a receber uma mensagem (o próprio processo) e repassá-la para o *process manager*, através de um `_sys_call`, que especifica de onde a *call* esta sendo feita (MM) e qual função da array esta sendo invocada.

Em `servers/pm/misc.c`, estão implementadas as funções `do_batch()` e `do_unbatch()` que utilizam a *systask* `sys_nice()` para mudar a prioridade do processo passado como argumento. Quando as chamadas dão certo, elas retornam `0` e quando dão errado, retornam `-1`, com isso, o usuario possui um *feedback* sobre a execução das chamadas implementadas. Além disso, checamos se o pid passado é válido (existe um processo com dado PID) e se foi o processo pai que realizou a tal chamada de sistema.

Em `kernel/systen/do_nice.c`, fizemos uma exceção para quando a nova prioridade é *BATCH_Q*, pois essa prioridade esta dfinida fora dos limites da prioridade do usuario, e a função normarizaria a prioridade para *caber* dentro das prioridades de um precsso de usuario.

4. O algoritmo de escalonamento

Em `kernel/proc.c`, modificamos o algoritmo de escalonamento. Em `sched()`, verificamos se o processo esta na fila de prioridade BATCH, então chamamos a função `batch()` para realizar o escalonamento especial deste tipo de processo. Caso contrário, a função `sched` termina de ser executada, sendo que a prioridade mais baixa é dada por BATCH_Q-1.

Consequentemente, todo processo que entra na função `batch()` esta na fila `BATCH_Q`. Garantimos que este

processo não muda, para isso, não alteramos a prioridade do mesmo. Em um laço, percorremos a fila BATCH procurando pelo menor e maior número de *tiques*. Se estes valores sejam iguais, sabemos que todos os processos em `BATCH_Q` têm o mesmo número de *tiques*, então o escalonador executa o *round robin*. Caso contrario, decidimos se o processo será colocado na frente (deve rodar) ou atrás (não deve rodar) da fila, levando em conta o número mínimo de *tiques* que algum processo em `BATCH_Q` possui e a quantidade de **tiques** que o processo sendo escalonado possui.

Em `kernel/proc.c` também editamos a função `balance_queues()` para garantir que a prioridade de um processo BATCH_Q não aumente, ou seja, garantir que nenhum processo saia de `BATCH_Q`.

Arquivos modificados

Aqui, resumimos quais arquivos foram modificados e o que foi modificado em cada um.

Arquivo	Mudança
`/usr/src/kernel/proc.h`	Nesse arquivo definimos o novo macro para a fila `BATCH`.
`/usr/src/kernel/proc.c`	Onde mudamos o escalonador. Basicamente fizemos um tratamento totalmente diferente para processos que estão com a prioridade definida como `BATCH`. Também alteramos a função `balance_queues`.
`/usr/src/kernel/system/do_nice.c`	Uma exceção foi adicionada para o caso em que a *system task* é invocada para mudar a prioridade de um processo para `BATCH_Q`
`/usr/src/servers/pm/table.c`	Adicionamos novas entradas no vetor, que representam as novas chamadas de sistama `do_batch` e `do_unbatch`.
`/usr/src/servers/pm/proto.h`	Prototipamos as novas chamadas.
`/usr/src/servers/pm/misc.c`	Fizemos aqui a definição das chamadas, ou seja, o código/maquinário das duas funções fica nesse arquivo.
`/usr/src/include/minix/callnr.h`	Definimos as macros `BATCH` e `UNBATCH`, a posição das novas chamadas no vetor.
`/usr/src/lib/posix/_batch.c`	Arquivo que vai realizar a *system call* relacionada à `BATCH`. É essa função que é chamada quando um programa de usuário invoca `batch(pid)`.
`/usr/src/lib/posix/_unbatch.c`	Arquivo que vai realizar a *system call* relacionada à `UNBATCH`. É essa função que é chamada quando um programa de usuário invoca `unatch(pid)`.

Para executar

Para construir no sistema as nossas modificações, fizemos, basicamente, duas sequências de make:

- Compilando a biblioteca/chamadas de sistema

```
# cd /usr/src/servers
# make image && make install
# cd /usr/src/lib/posix
# make Makefile
# cd /usr/src
# make libraries
```

- Compilando o kernel e criando nova imagem de boot

```
# cd /usr/src/tools
# make hdboot && make install
```

Após isso, foi necessario dar um *reboot* no sistema com a nova imagem.

Testando

Para testarmos nosso escalonador, criamos o arquivo *teste.c* na *home*, que possui um processo que roda em baixíssima prioridade, chamando `batch()` e `unbatch()` para esse processo. Rodamos esse arquivo em background e executamos o comando `top`. Assim é possível ver a mudança de prioridade do processo *teste*, comprovando que o nosso escalonador funciona.