MAC0422 - SISTEMAS OPERACIONAIS

Relatório EP2

Washington Luiz - 10737157

Ygor Tavela Alves - 10687642

1. Prioridade BATCH_Q

A macro *BATCH_Q* foi adicionada entre o *IDLE* e as filas de usuário. Para isso, aumentamos o número de filas (*NR_SCHED_QUEUES*) para **17**, assim a fila *IDLE* foi definida como **16** e a fila *BATCH_Q* corresponde à fila **15**. Essa mudança foi realizada em *usr/src/kernel/proc.h* .

2 && 3. *System calls batch* e *unbatch*

Para fazer o **item 2** e **3**, ou seja, implementar as chamadas de sistema *batch* e *unbatch*, foi necessário implementar uma *chamada de kernel* (system task *setprior*) auxiliar. Abaixo descrevemos o *workflow* delas.

Chamada de sistema **batch**/**unbatch**:

- Parâmetros: **PID** do processo que deve entrar/sair na fila **BATCH_Q**;
- Rotina:
    1. Em *src/lib/posix/[_batch.c | _unbatch]* (com protótipo em header *unistd.h*) é criado uma nova *message* **m**, contendo o **PID** do processo e o **PID** do processo de quem a invocou;
    2. Envia **m** ao *process manager* (**MM**), indicando qual a rotina (**BATCH**/**UNBATCH**) deve ser chamada: ***_syscall(MM, BATCH/UNBATCH, &m)***. Sendo as macros **BATCH**/**UNBATCH** definidas em *src/include/minix/callnr.h* e, mapeadas na tabela de rotinas em *src/server/pm/table.c*, cujas rotinas correspondentes são **do_batch**/**do_unbatch**;
    3. Chama a rotina **do_batch**/**do_unbatch** (*src/server/pm/misc.c*) que extrai os parâmetros de **m** e, verifica se o processo que a chamou é **pai** do processo (**filho**) que deve entar/sair na fila **BATCH_Q**. Se for o caso, é feito uma *chamada de kernel* **sys_setprior**, parametrizada com o número do processo **filho** na tabela de processos e com a nova fila de prioridade, a qual, o processo deve entrar.
- Obs.: a diferença de batch pra unbatch é que a primeira coloca na fila BATCH_Q e a segunda tira da BATCH_Q , colocando na fila USER_Q.

Chamada de kernel **sys_setprior**:

- Parâmetros: o número do processo e a nova fila prioridade;
- Rotina:
    1. Em *src/lib/syslib/sys_setprior.c* é passado os parâmetros para o ponteiro *message* **m**;
    2. Envia **m** para o *system task* (**SYSTASK**), indicando qual rotina (**SYS_SETPRIOR**) a ser chamada: ***_taskcall(SYSTASK, SYS_SETPRIOR, &m)***. Sendo a macro **SYS_SETPRIOR**

definida em *src/include/minix/com.h* e, mapeada em *src/kernel/system.c*;
3. Chama a rotina **do_setprior** (*src/kernel/system/do_setprior.c*). Ela extrai os parâmetros da mensagem **m**, o número do processo (**proc_nr**) e sua prioriade (**pri**). Com o número do processo recuperamos um apontador para processo (**rp**) atráves da rotina **proc_addr**. Finalmente, chamados a função **lock_dequeue**, a qual tira o processo da fila, bloqueando ele, e atualizamos a prioriade do processo, no caso, para **prio**. Assim, chamamos a função **lock_enqueue**, a qual coloca o o processo na fila correspondente à **prio**.

4. Mudanças na política de scheduling

A seguir, introduzimos as mudanças realizadas para cada item pedido. Todas as mudanças foram feitas no arquivo **proc.c**.

1. "Nenhum processo em BATCH_Q muda de fila"
    - Para garantir essa condição, tivemos que modificar a rotina **balance_queues**. Na linha *707*, na qual é verificado a necessidade de atualizar a prioridade de um processo, adicionamos mais uma condição lógica para não permitir que os processos com prioridade igual a *BATCH_Q* seja atualizado;
    - Além de nenhum processo poder sair da *BATCH_Q*, nós garantimos que nenhum processo pode entrar na mesma. Para isso, na linha *645* da rotina **sched**, diminuimos a prioridade máxima que um processo pode ter, ou seja, *IDLE_Q - 2*.
2. "Um processo novo em BATCH_Q deve rodar até que o seu total de tiques seja o mesmo do processo com menor número de tiques na fila"
    - Dentro da rotina **sched**, caso o processo a ser escalonado tenha prioriade equivalente a *BATCH_Q*, percorremos a fila de prioridade para encontrar o menor valor de *p_ticks_left* dentre os processos, atribuindo à *min_ticks_left* este valor. Desta forma, se o processo a ser escalonado tem seu número de *p_ticks_left* menor que o *min_ticks_left* calculado, esse processo é colocado no final da fila, caso contrário, ele se manterá na frente da fila mantendo a sua execução. Tal procedimento pode ser verificado entre as linhas *613* à *628*, onde a linha *656* é responsável por manter ou alterar a posição na fila de prioridade da BATCH_Q.
3. "Quando todos processos de BATCH_Q tiverem o mesmo número de tiques, os processos são escalonados em round robin"
    - Criamos uma variável booleana *check_cond_three* em **sched**, que inicialmente é verdadeira, dentro do *for* utilizado no item **4.2** para percorrer a fila *BATCH_Q*. Verificamos se um processo possui sua quantidade de *p_ticks_left* menor que o menor valor de ticks *min_ticks_left*, ou seja, caso seja verdade tal afirmação então os processos da fila não se encontram em uma situação tal que todos eles possuem o mesmo número de tiques, logo na linha *619*, o estado da variável *check_cond_three* é alterado para falso. Além disso, no mesmo *for* realizamos a busca pelo processo que possuí o menor *p_quantum_size*, sendo tal informação armazenada em *min_quantum_size*.
    - Caso a condição do item **4.2** seja verificada, ou seja, o processo a ser escalonado tenha uma quantidade de ticks igual a do processo com menor número de ticks na fila, então, tratamos o processo como um processo sem *time_left* (todo o quantum foi consumido). Assim, nas linhas *639* e *640*, caso a flag *check_cond_three* seja verdadeira atribuimos um novo quantum ao processo que é igual ao valor mínimo de quantum dado pela variável *min_quantum_size*.
    - Por fim, caso o processo da fila *BATCH_Q* tenha "esgotado" o seu quantum (*time_left*), a posição que ele irá assumir na fila será dada pela flag *check_cond_three* na linha *656*.
4. "Processos nesta fila só rodam quando a máquina está ociosa."

- Da forma que escolhemos implementar a fila *BATCH_Q* no item **1**, ou seja, atribuindo ela a posição entre a fila *IDLE* e as filas de usuário. Por padrão a função **pick_proc**, irá escolher um processo da fila *BATCH_Q* apenas se não houver processos em filas de prioridade maior que ela. Desta forma, um processo nesta fila só irá rodar quando, teoricamente, não houver mais nenhum processo além do *IDLE*.