INE 5412 – Sistemas Operacionais I

Trabalho Prático 1: Escalonamento de Processos

Márcio Castro, Fernando Mota e Pedro Penna

Resumo

Processos consistem na abstração fundamental de um sistema operacional: todo o resto do sistema é arquitetado com base nesta entidade. Neste projeto, você trabalhará com o principal componente do módulo de gerenciamento de processos do Nanvix, o escalonador de processos. Primeiramente, você irá estudar como o componente existente opera e, em seguida, irá propor melhorias a ele.

Fundamentação Teórica

O Nanvix é um sistema operacional multitarefa, isto é, ele suporta a execução simultânea de diferentes programas. O modo como esta funcionalidade é proporcionada, tanto no Nanvix quanto em outros sistemas, é simples. Cada programa é abstraído por uma entidade denominada processo, que encapsula o fluxo de execução, as variáveis, a pilha de execução e várias outras informações importantes relacionadas ao programa. Então, com esta abstração, tudo o que o sistema faz é alternar rapidamente entre as tarefas, proporcionando a ilusão de simultaneidade.

O procedimento de alternar tarefas em si é direto, envolve apenas recarregar os registradores de máquina e as variáveis de ambiente com os dados do processo sendo admitido para execução. A etapa crítica, no entanto, está na escolha do processo a ser admitido para execução. Com vários processos residentes no sistema, diversas escolhas são possíveis: admitir o processo que estiver aguardando há mais tempo, admitir o processo que precisará do menor tempo para terminar, ou então admitir o processo com a mais alta prioridade, segundo um critério pré-estabelecido. Essa escolha é feita pelo módulo do sistema denominado escalonador de processos e exerce influência direta no desempenho, responsividade e política de justiça do sistema.

O escalonador de processos do Nanvix adota a simples política *round-robin*: atribuir a cada processo um *quantum* de tempo e então os escalona seguindo o critério *first-in first-out*. Para tanto, o sistema mantém, para cada processo, um contador que indica o tempo em que um processo está aguardando para executar. Quando o *quantum* do processo executando acaba, o processo com maior contador é selecionado, atribuído um novo *quantum* e, então, é colocado para executar.

Essa política, de fácil implementação, é adequada para sistemas monotarefa simples, mas é insuficiente para sistemas mais robustos, com suporte a multitarefa, como o Nanvix. Suponha uma situação em que exista um processo que interage com o usuário, por exemplo um terminal, e outros 99 processos famintos por tempo de processador. Se o sistema atribui a cada processo 100 ms de *quantum*, cada processo executará a cada dez segundos. Apesar de plausível para os 99 processos limitados pelo tempo de processamento, esse tempo de espera para o processo de terminal é inadmissível.

Descrição do Projeto

Você deverá implementar um escalonador **inspirado** no ***Completely Fair Scheduler (CFS)***[1] do Linux. O CFS é baseado na ideia de um processador multitarefa ideal. Tal processador seria capaz de executar todos os processos ativos literalmente ao mesmo tempo, onde cada processo teria uma porção do poder

[1] Mais informações sobre o CFS podem ser encontradas em `https://www.ibm.com/developerworks/library/l-completely-fair-scheduler/index.html` e `https://www.prism-services.io/pdf/linux_scheduler_notes_final.pdf`

de processamento do processador. Por exemplo, considere um sistema com somente 2 processos. Em um processador multitarefa ideal, os 2 processos poderiam ser executados ao mesmo tempo, utilizando cada um 50% do poder de processamento do processador. Assumindo que os 2 processos tenham iniciado suas execuções ao **mesmo tempo** e que ambos possuam a **mesma prioridade**, o tempo de execução deles no processador em qualquer momento seria **exatamente o mesmo**. Portanto, esta seria uma situação completamente justa.

Como **não é fisicamente possível que os processadores atuais operem de tal forma**, o CFS tenta aproximar esse comportamento o máximo possível. Com base em um tempo execução virtual de cada processo (`vruntime`), o CFS tenta manter um balanço geral entre todos processos em execução, ou seja, tenta garantir que o `vruntime` de todos os processos seja na média muito próximo. Em cada chamada do escalonador, o `vruntime` de um processo é incrementado de acordo com tempo em que o processo utilizou o processador. O processo sofrerá uma preempção quando o seu `vruntime` for maior que o menor `vruntime` dos processos no estado pronto (`min_vruntime`). Para garantir que novos processos possam ser escalonados rapidamente, o `vruntime` de um processo recém criado é setado para `min_vruntime`.

Para implementar esta ideia você deverá utilizar um **tempo de base** (*target scheduling latency* - TSL) para calcular o tempo a ser dado à cada processo. Por exemplo, assumindo-se um TSL de 20ms e somente 2 processos de mesma prioridade, cada processo seria executado durante 10ms até ser preemptado em favor do outro. Note que esta parcela de tempo poderá se tornar muito pequena se o sistema tiver muitos processos ativos, resultando em um grande sobrecusto em trocas de contexto. Portanto, o TSL deverá ser ajustado em função do número de processos ativos no sistema com base em uma granularidade mínima definida (`sched_min_granularity`).

Por fim, tenha em mente que a prioridade estática dos processos (atributo `priority` do processo no Nanvix) e de usuário (atributo `nice` do processo no Nanvix) deverão ser consideradas como um fator para acelerar ou diminuir o incremento do `vruntime`, aumentando-se assim a chance de escalonar processos mais prioritários em um curto espaço de tempo.

Neste trabalho, você deverá se ater ao essencial do CFS, não sendo necessário implementar os conceitos *classes*, *groups* e *domains*. Junto com o projeto você deverá preparar uma apresentação da solução proposta na forma de slides, os quais deverão ser apresentados em uma data prevista no cronograma da disciplina. Além de explicar a solução implementada, você deverá indicar quais são as vantagens e (possíveis) desvantagens da estratégia projetada.

BÔNUS!

Como é possível perceber, o algoritmo do CFS depende bastante da "velocidade" com que é possível obter o processo com menor `vruntime` para poder ser eficiente. Da mesma forma, o algoritmo também depende de uma forma de inserir e atualizar esses valores de maneira ordenada para poder sempre manter o processo com menor valor à disposição. Nesse sentido, uma árvore binária de busca pode ser utilizada para manter uma ordenação dos processos de acordo com os seus `vruntime`, reduzindo assim o tempo de busca. Em especial, um tipo específico de árvore binária de busca denominada árvore rubro-negra apresenta resultados ainda melhores. Atualmente a versão do CFS implementada no kernel do Linux utiliza uma árvore rubro-negra, permitindo a inserção e deleção de processos em $O(\log n)$, enquanto possibilita acesso ao processo com menor `vruntime` em $O(\log n)$. **Você será recompensado com 2 pontos na P1 se a sua solução estiver correta e fizer uso de uma árvore rubro-negra**.

Por Onde Começar?

O código do gerenciador de processos do Nanvix está no diretório `kernel/pm`, dividido em vários arquivos, dentre eles:

- `pm.c`: inicialização do gerenciador de processos.
- `sched.c`: escalonamento de processos.

- `sleep.c`: implementação das funções utilitárias `sleep()` e `wakeup()`.

Além do módulo de gerenciamento de processos, os seguintes arquivos também poderão ser úteis:

- `src/kernel/arch/x86/clock.c`: temporizador de interrupções.
- `src/kernel/sys/fork.c`: chamada de sistema para criação de processos.

Você irá notar que os processos no Nanvix já possuem atributos relacionados à prioridade, os quais são atualmente ignorados pelo escalonador: `priority` e `nice`. Pesquise no código fonte do Nanvix e em outras fontes para entender o funcionamento delas.

Testes Básicos

O Nanvix possui um programa que permite realizar alguns testes de escalonamento, entrada/saída, comunicação entre processos e paginação denominado `test`. O seu código fonte está localizado no diretório `src/sbin/test`.

Para realizar os testes de escalonamento execute `test sched` a partir do terminal do Nanvix. A execução deverá demorar um certo tempo. Esse programa utilitário realizará alguns testes de stress envolvendo o escalonador de processos do kernel. Caso você sinta necessidade de exercitar outras funcionalidades do seu algoritmo, fique à vontade para modificar esse utilitário.

Grupos, Avaliação e Entrega

O trabalho deverá ser realizado necessariamente em grupos. Para entrar em algum grupo, acesse o *link* do *Github Classroom* desta atividade disponível no Moodle.

ATENÇÃO!
Não será permitida a criação de grupos no Github Classroom. Trabalhos que não forem realizados em algum dos grupos disponibilizados pelo professor **não serão avaliados**.

Os trabalhos serão apresentados nos dias definidos no cronograma disponível no Moodle. O professor irá avaliar tanto a **corretude**, **desempenho** e **clareza** da solução. A data/hora limite para o envio dos trabalhos é **28/04/2019 às 23h59min**. Não será permitida a entrega de trabalhos fora desse prazo.

Durante a apresentação, o professor irá avaliar o **conhecimento individual dos alunos sobre os conteúdos teóricos e práticos vistos em aula e sobre a solução adotada no trabalho**. A nota atribuída à cada aluno i no trabalho ($NotaTrabalho_i$) será calculada da seguinte forma, onde A_i é a nota referente à apresentação do aluno i e S é a nota atribuída à solução do trabalho:

$$NotaTrabalho_i = \frac{A_i * S}{10} \tag{1}$$

ATENÇÃO!
Como indicado pela fórmula mostrada acima, **a nota atribuída à solução adotada será ponderada pelo desempenho do aluno durante a apresentação do trabalho**. Por exemplo, se o professor atribuir nota 10 para a solução adotada pelo grupo mas o aluno receber nota 5 pela apresentação – devido ao desconhecimento dos conteúdos teóricos, práticos e/ou da solução do trabalho – a sua nota final do trabalho será 5. A ausência no dia da apresentação ou recusa de realização da apresentação do trabalho implicará em nota zero na apresentação, fazendo com que a nota atribuída ao aluno também seja zero.