Universidade Federal de Viçosa – Campus UFV-Florestal
Ciência da Computação – Sistemas Operacionais
Professor: Daniel Mendes Barbosa

Trabalho Prático 2 – Data de entrega: ver no Google Classroom

Este trabalho prático deverá ser feito em grupos de 4 ou 3 alunos, conforme definido com o professor (teremos 3 grupos de 4 alunos e 3 grupos de 3 alunos). Deverá ser entregue um **relatório** através do Google Classroom, contendo uma **breve documentação e decisões importantes do projeto (um PDF, como sempre, não zipado)**, além de testes de execução com *screenshots*. Deverá ser entregue também todo o código do projeto (em um único arquivo .zip, entregue em local específico no Google Classroom), implementado na linguagem C e no sistema operacional Linux. Outras linguagens de programação podem ser usadas, mas deve haver uma primeira seção da documentação explicando qual ou quais foram usadas, e as justificativas, além de relatar os recursos utilizados destas linguagens. Esse trabalho também deverá ser **apresentado** em aula, mostrando essas principais decisões de projeto e o seu funcionamento com **exemplos significativos, diversos e bem justificados**. Qualquer diferença entre o que for implementado e a especificação abaixo deve constar claramente na documentação e na apresentação e ser justificada. É suficiente que apenas um aluno do grupo faça a entrega, colocando nos nomes dos alunos no PDF da documentação.

Trabalho: simulação de gerenciamento de processos

Objetivo: Simular cinco funções do gerenciamento de processos: criar processo, substituir a imagem atual do processo com uma imagem nova do processo, transição de estado do processo, escalonamento de processos e troca de contexto.

Você utilizará chamadas de sistema de Linux como *fork ( ), wait ( ), pipe ( ) e sleep ( )*. Leia as páginas man dessas chamadas de sistema para detalhes. Veja exemplos de fork aqui.

Esta simulação consiste de três tipos de processo em Linux: **controle, gerenciador de processos** e **impressão**. Existe um processo **controle** (esse é o processo que inicia a sua simulação), um processo **gerenciador de processos** que é criado pelo processo **controle** e uma série de processos **impressão** que são criados por **gerenciador de processos**, conforme necessário.

Processo controle:

O processo controle primeiro cria um pipe e depois um processo gerenciador de processos. Depois ele lê comandos repetidamente a partir da entrada padrão ou a partir de um arquivo (o usuário do seu simulador irá escolher esta opção inicialmente) e passa-os para gerenciador de processos através do pipe. Existem quatro tipos de comandos:

1. U: Fim de uma unidade de tempo.
2. L: Desbloqueia o primeiro processo simulado na fila bloqueada.
3. I: Imprime o estado atual do sistema.
4. M: Imprime o tempo médio do ciclo e finaliza o sistema.

Comando M aparece exatamente uma vez, esse é o último comando.

Processo simulado:

A simulação de gerenciamento de processo gerencia a execução de processos simulados. Cada processo simulado é composto de um programa que manipula (define/atualiza) o valor de variáveis inteiras. Dessa forma, o estado de um processo simulado, a qualquer momento, é composto pelos valores da suas variáveis inteiras e pelo valor do seu contador de programa. O programa de simulação de processo consiste de uma sequência de instruções. Existem os oito tipos de instruções, conforme segue:

1. **`N n`**: número de variáveis que serão declaradas neste processo simulado.
2. **`D x`**: declara uma variável inteira x (x será uma referência simplificada à memória, sendo um número inteiro começando do zero), alocando espaço para ela em um vetor representando a memória (o valor de x seria usado então para indexar o vetor). Cada processo simulado terá o seu vetor, simulando a sua memória para as suas estruturas de dados (suas variáveis). Ao declarar uma variável, seu valor inicial deverá ser zero.
3. **`V x n`**: Define o valor da variável inteira x para n, onde n é um inteiro.
4. **`A x n`**: Adiciona n ao valor da variável inteira x, onde n é um inteiro.
5. **`S x n`**: Subtrai n do valor da variável inteira x, onde n é um inteiro.
6. **`B`**: Bloqueia esse processo simulado.
7. **`T`**: Termina esse processo simulado.
8. **`F n`**: Cria um novo processo simulado. O novo processo (simulado) é uma cópia exata do processo pai (simulado). O novo processo (simulado) executa a partir da instrução, imediatamente após essa instrução F, enquanto o processo pai (simulado) continua a sua execução n instruções, depois da próxima instrução.
9. **`R nome_do_arquivo`**: Substitui o programa do processo simulado pelo programa no arquivo nome_do_arquivo e define o contador de programa para a primeira instrução desse novo programa.

Um exemplo de um programa para um processo simulado é mostrado a seguir:

```
N 2
D 0
D 1
V 0 1000
V 1 500
A 0 19
A 0 20
S 1 53
```

```
A 1 55
F 1
R file_a
F 1
R file_b
F 1
R file_c
F 1
R file_d
F 1
R file_e
T
```

Você pode armazenar o programa de um processo simulado também em um vetor, com cada elemento do vetor contendo uma instrução.

Processo gerenciador de processos:

O processo gerenciador de processos simula cinco funções de gerenciamento de processos: criar um novo processo (simulado), substituir a imagem atual de um processo simulado por uma imagem nova de processo, gerenciar a transição de estados do processo, escalonar processos e trocar contexto. Além disso, ele inicia o processo impressão sempre que precisa imprimir o estado do sistema.

O gerenciador de processos cria o primeiro processo simulado (process id = 0). O programa para esse processo é lido a partir de um arquivo (nome_do_arquivo: init). Este é o único processo simulado criado pelo gerenciador de processos. Todos os outros processos simulados são criados em resposta à execução da instrução F.

Gerenciador de processos: estruturas de dados

O Gerenciador de processos mantém seis estruturas de dados: Tempo, Cpu, TabelaDeProcessos, EstadoPronto, EstadoBloqueado e EstadoExecução. Tempo é um valor inteiro inicializado para zero. Cpu é usado para simular a execução de um processo simulado que está em estado de execução. Deve incluir membros de dados para armazenar um ponteiro para o vetor de programa, valor atual de contador de programa, e ponteiro para o vetor que simula a memória das estruturas de dados e fatia de tempo desse processo simulado. Ou seja, é como se esta estrutura possuísse registradores para armazenar estes dados, e enquanto estiver em execução, tudo deve ser acessado a partir destes registradores, e só depois, numa troca de contexto, é que deveria ser copiado de volta às estruturas do processo e da TabelaDeProcessos. Além disso, ele deve guardar o número de unidades de tempo usadas, até então, na fatia de tempo atual.

TabelaDeProcessos é um vetor com uma entrada para cada processo simulado que ainda não terminou a sua execução. Cada entrada deve incluir membros de dados para armazenar identificador do processo, identificador do processo pai, um ponteiro para o

valor de contador de programa (inicialmente 0), ponteiro para o vetor de estruturas de dados, prioridade, estado, tempo de início e tempo de CPU, usados, até então.

EstadoPronto armazena todos os processos simulados (índices de TabelaDeProcessos) que estão prontos para executar. Isso pode ser implementado usando uma estrutura de dados de uma fila ou de uma fila de prioridades. EstadoBloqueado armazena todos os processos (índices de TabelaDeProcessos) que estão bloqueados no momento. Isso pode ser implementado com uma estrutura de dados de uma fila. Finalmente, EstadoExecução armazena o índice de TabelaDeProcessos do processo simulado, atualmente em execução.

Gerenciador de processos: processando comandos de entrada

Após criar o primeiro processo e inicializar todas as suas estruturas de dados, o gerenciador de processos recebe, repetidamente, e processa um comando por vez, a partir do processo controle (leitura através do pipe). Ao receber um comando U, o gerenciador de processos executa a próxima instrução do processo simulado, atualmente em execução, incrementa o valor do contador de programa (exceto para instruções F ou R), incrementa Tempo e depois faz o escalonamento. Observe que o escalonamento pode envolver a troca de contexto.

Ao receber um comando L, gerenciador de processos move o primeiro processo simulado da fila bloqueada para a fila de estado pronto para executar. Ao receber um comando I, gerenciador de processos dispara um novo processo impressão. Ao receber um comando M, gerenciador de processos primeiro dispara um processo impressão e depois termina após a finalização do processo impressão. gerenciador de processos garante que apenas um processo impressão execute ao mesmo tempo.

Gerenciador de processos: executando processos simulados

O gerenciador de processos executa a próxima instrução do processo simulado, atualmente em execução, ao receber um comando U, a partir do processo controle. Observe que essa execução é totalmente confinada à estrutura de dados Cpu, isto é, TabelaDeProcessos não é acessada.

Instruções D, V, A e S atualizam os valores inteiros das variáveis armazenados em Cpu. Instrução B move o processo simulado, atualmente em execução, para o estado bloqueado e move um processo do estado pronto para o estado em execução. Isso resultará em uma troca de contexto. Instrução T finaliza o processo simulado, atualmente em execução, libera toda a memória (por exemplo, vetor de programa) associada a esse processo e atualiza TabelaDeProcessos. Um processo simulado é movido do estado pronto para o estado em execução. Isso também resulta em uma troca de contexto.

Instrução F resulta em criação de um novo processo simulado. Um novo identificador (único) é atribuído e o identificador do processo pai é o identificador de processo do processo pai simulado. Tempo de início é definido para tempo atual e o tempo de CPU

usado, até então, é ajustado para 0. O vetor de programa e o vetor de valores inteiros do novo processo simulado são uma cópia do vetor de programa e do vetor de valores inteiros do processo pai simulado. O novo processo simulado possui a mesma prioridade que a do processo pai simulado. O valor do contador de programa do novo processo simulado é ajustado para a instrução imediatamente após a instrução F, enquanto o valor de contador de programa do processo pai simulado é ajustado para n instruções, depois da próxima instrução (instrução imediatamente depois da F). O novo processo simulado é criado com estado pronto.

Finalmente, a instrução R resulta em substituir a imagem do processo simulado, atualmente em execução. Seu vetor de programa é sobrescrito pelo código no arquivo nome_do_arquivo, o valor de contador de programa é definido para 0 e o vetor de valores inteiros será substituído pelo que for definido no arquivo. Observe que todas essas mudanças são feitas, apenas na estrutura de dados Cpu. Identificador de processo, identificador de processo pai, tempo de início, tempo de CPU usado até então, estado e prioridade permanecem inalterados.

Gerenciador de processos: escalonamento

O gerenciador de processos também implementa uma política de escalonamento. Você pode fazer experiências com política de escalonamento de múltiplas filas com classes de prioridade. Nessa política, o primeiro processo simulado (criado pelo gerenciador de processos) inicia com a prioridade 0 (prioridade mais alta). Existem no máximo quatro classes de prioridades. Fatia de tempo (tamanho de quantum) para a classe de prioridade 0 é uma unidade de tempo; fatia de tempo para a classe de prioridade 1 são duas unidades de tempo; fatia de tempo para a classe de prioridade 2 são quatro unidades de tempo; e fatia de tempo para a classe de prioridade 3 são oito unidades de tempo. Note que os processos com altas prioridades devem ser executados primeiro, mas não podem roubar muito o tempo da CPU e portanto precisam ter um menor tempo de execução. Se o processo em execução usar a sua fatia de tempo por completo, a sua prioridade é diminuída. Note que se um processo com alta prioridade acabar com sua fatia de tempo, o mesmo ganhará um tempo maior para executar, mas terá uma prioridade menor com relação à ordem de escalonamento. Portanto, veja que os processos estão dando o lugar um para o outro de uma forma justa baseada em prioridades. Se o processo for bloqueado, antes de expirar seu quantum alocado, a sua prioridade é aumentada.

Gerenciador de processos: política de escalonamento do grupo

Além da política de escalonamento descrita acima, o grupo deverá implementar uma outra política, baseada no que foi estudado na disciplina. Esta política deverá ser descrita no relatório de documentação em alto nível e como foi implementada. É um ponto importante também para a apresentação do trabalho.

Gerenciador de processos: troca de contexto

Troca de contexto envolve copiar o estado do processo simulado, atualmente em execução, de Cpu para TabelaDeProcessos (a não ser que esse processo tenha completado a sua execução) e copiar o estado do recém escalonado processo simulado de TabelaDeProcessos para Cpu.

Processo impressão

O processo impressão imprime o estado atual do sistema para saída padrão e depois pode finalizar (finalizar somente no caso de ser um comando M do processo controle). A saída do processo impressão **deve ser construída pelo grupo**, devendo trazer o máximo de informações sobre esta simulação, como dados dos processos, em que estado eles estão, em que fila estão, entre outros, mas organizados de uma maneira de fácil leitura. Podem haver também mais de uma configuração, de forma que as informações impressas sejam diferentes, ou mais detalhadas ou mais simplificadas.

Possíveis complementos à especificação acima:

1) O grupo poderá melhorar a interface do simulador criado, desde que especifique estas melhorias no relatório e na apresentação e mostre seus resultados/vantagens.

2) O uso de processos reais no simulador como descrito também pode ser melhorado/alterado, podem ser usados threads, semáforos, monitores, desde que tudo seja explicado e justificado. Ou seja, o ideal é que a implementação com processos reais deve ser feita. Estas outras implementações seriam extras, apenas para melhorar o trabalho.

3) A especificação considera uma máquina com apenas uma CPU, ou seja, apenas um processo em execução em determinado momento. O grupo poderá implementar 2 ou mais núcleos também explicando e justificando as decisões para tal implementação. Outra possibilidade é a configuração no momento de se iniciar a simulação, de quantos núcleos estarão disponíveis.