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

Trabalho Prático 2 – Data de entrega: ver PVANet

Este trabalho prático deverá ser feito em grupos de 4 alunos (haverá apenas um grupo de 5 alunos). Deverá ser entregue um **relatório** através do PVANet, contendo uma **breve documentação e decisões importantes do projeto**, além de testes de execução com *screenshots*. Deverá ser entregue também todo o código do projeto, implementado na linguagem C e no sistema operacional Linux. 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.

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 e wait em: http://www.dei.isep.ipp.pt/~andre/documentos/multiprocesso.html

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

Processo commander:

O processo commander primeiro cria um pipe e depois um processo process manager. 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 process manager através do pipe. Existem quatro tipos de comandos:

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

Comando T 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 uma única variável inteira. Dessa forma, o estado de um processo simulado, a qualquer momento, é composto pelo valor da sua variável inteira e pelo valor do seu contador de programa. O programa de simulação de processo consiste de uma sequência de instruções. Existem sete tipos de instruções, conforme segue:

1. **S n**: Define o valor da variável inteira para n, onde n é um inteiro.
2. **A n**: Adiciona n ao valor da variável inteira, onde n é um inteiro.
3. **D n**: Subtrai n do valor da variável inteira, onde n é um inteiro.
4. **B**: Bloqueia esse processo simulado.
5. **E**: Termina esse processo simulado.
6. **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.
7. **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:

```
S 1000
A 19
A 20
D 53
A 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
E
```

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

Processo process manager:

O processo process manager 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 reporter sempre que precisa imprimir o estado do sistema.

Process manager 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 process manager. Todos os outros processos simulados são criados em resposta à execução da instrução F.

Process manager: estruturas de dados

Process manager mantém seis estruturas de dados: Time, Cpu, PcbTable, ReadyState, BlockedState e RunningState. Time 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, valor inteiro e fatia de tempo desse processo simulado. Além disso, ele deve guardar o número de unidades de tempo usadas, até então, na fatia de tempo atual.

PcbTable 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), valor inteiro, prioridade, estado, tempo de início e tempo de CPU, usados, até então.

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

Process manager: processando comandos de entrada

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

Ao receber um comando U, process manager move o primeiro processo simulado da fila bloqueada para a fila de estado pronto para executar. Ao receber um comando P, process manager dispara um novo processo reporter. Ao receber um comando T, process manager primeiro dispara um processo reporter e depois termina após a finalização do processo reporter. Process manager garante que apenas um processo reporter execute ao mesmo tempo.

Process manager: executando processos simulados

Process manager executa a próxima instrução do processo simulado, atualmente em execução, ao receber um comando Q, a partir do processo commander. Observe que essa execução é totalmente confinada à estrutura de dados Cpu, isto é, PcbTable não é acessada.

Instruções S, A e D atualizam o valor inteiro armazenado 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 E finaliza o processo simulado, atualmente em execução, libera toda a memória (por exemplo, vetor de programa) associada a esse processo e atualiza PcbTable. 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 valor inteiro do novo processo simulado são uma cópia do vetor de programa e do valor inteiro 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 valor inteiro é indefinido. 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.

Process manager: escalonamento

Process manager 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 process manager) 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. Se o processo em execução usar a sua fatia de tempo por completo, a sua prioridade é diminuída. Se o processo for bloqueado, antes de expirar seu quantum alocado, a sua prioridade é aumentada.

Process manager: 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.

Process manager: troca de contexto

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

Processo reporter

O processo reporter imprime o estado atual do sistema para saída padrão e depois pode finalizar (finalizar somente no caso de ser um comando T do processo Commander). A saída do processo reporter é semelhante a:

```
**********************************************************************
The current system state is as follows:
**********************************************************************
CURRENT TIME: time
RUNNING PROCESS:
pid, ppid, priority, value, start time, CPU time used so far
BLOCKED PROCESSES:
Queue of blocked processes:
pid, ppid, priority, value, start time, CPU time used so far
...
pid, ppid, priority, value, start time, CPU time used so far
PROCESSES READY TO EXECUTE:
Queue of processes with priority 0:
pid, ppid, value, start time, CPU time used so far
pid, ppid, value, start time, CPU time used so far
...
...
Queue of processes with priority 3:
pid, ppid, value, start time, CPU time used so far
pid, ppid, value, start time, CPU time used so far
**********************************************************************
```

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 grupo poderá usar outra linguagem de programação, de modo a complementar os resultados, ou mesmo usar outra linguagem de programação para fazer outra versão do trabalho, com vantagens/desvantagens. Esta outra versão pode ser entregue sozinha, ou então podem ser entregues as duas versões, observado o item 3, a seguir.

3) 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 deve ser feita. Estas outras implementações seriam extras, apenas para melhorar o trabalho. No entanto, se for uma implementação muito bem feita e que traga novidades interessantes ao trabalho, o grupo pode ainda conseguir nota máxima mesmo sem a implementação em C e com processos como foi pedido.

4) 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.