Pseudo SO - Trabalho Prático

Nome:
Alexandre Kaihara (180029690)
Higor Gabriel Azevedo Santos (170012387)
Davi Cunha Farias Dupin (170140148)
Pedro Luis Chaves Rocha (180054635)
Turma: Fundamentos de Sistemas Operacionais - Turma A

1) Descrição das ferramentas/linguagens usadas

A linguagem utilizada para esse trabalho foi Python. No trabalho foram utilizadas as bibliotecas “Threading” para a sincronização e garantir a exclusão mútua. A biblioteca “Queue” para implementar as filas de prioridade. A biblioteca “datetime” para obter os timestamps do programa. A biblioteca “Random” e “time” para poder simular um comportamento aleatório do sistema. E por fim a biblioteca “sys” para poder receber do terminal os caminhos para os arquivos de entrada para o programa.

2) Descrição teórica e prática da solução dada

O sistema foi subdividido em seis grandes módulos, sendo: Módulos de fila, processos, arquivos, memória e recurso, implementados de modo independente e integrados pelo último módulo chamado de PseudoOS, responsável por implementar toda a lógica de controle e interação entre todos os módulos.

Nas seções a seguir contém a descrição detalhada de cada módulo e suas respectivas soluções teóricas e práticas abordadas neste trabalho.

2.1) Módulo de arquivos

O módulo de arquivos é um segmento do código implementado que representa o Sistema de Arquivos do nosso pseudo-SO. Aqui realizamos todo o gerenciamento de arquivos no que tange à criação e à exclusão destes.

A estratégia utilizada para realizar essas operações segue todos os princípios pré-estabelecidos pela especificação do trabalho, isto é: há persistência dos dados em disco; a alocação é contígua; o algoritmo de alocação do disco é o first-fit; processos de tempo real podem criar e deletar qualquer arquivo; e processos comuns possuem a limitação de deletar apenas arquivos criados por eles.

A estrutura do sistema de arquivos está concentrada, a nível de código, em uma classe, que, a partir da necessidade de criação ou exclusão de um determinado arquivo, vai realizar a operação solicitada pelo Módulo do PseudoOS.

Para a criação de um arquivo, são necessários os parâmetros *processID, filename, filesize e priority*. Caso um arquivo de mesmo nome já tenha sido criado, é retornado um aviso invalidando a operação, caso contrário, é realizado o first-fit - itera a memória buscando o primeiro espaço vago que caiba o novo arquivo (vale ressaltar que o programa retorna erro caso não haja memória disponível).

De forma análoga, para deletar um arquivo, é preciso saber o *processID, filename* e o booleano *isRealTime*. Realizamos a validação para confirmar se o processo tem permissão para apagar aquele arquivo e, na mesma iteração, já é mapeado o tamanho completo do bloco de memória que o guarda para que os dados sejam deletados.

O módulo de arquivos, após realizar a operação demandada, retorna ao pseudo-OS o que foi realizado (se foi possível completar a operação com sucesso ou se houve erro) e também imprime no terminal o log das atividades feitas.

2.2) Módulo de filas

O módulo de filas tem a responsabilidade de gerenciar todas as 4 filas de prioridade que o sistema operacional possui (incluindo a de processos de tempo real). Além disso, também é utilizado uma estratégia de aging que visa aumentar a prioridade de um processo caso ele fique muito tempo na mesma fila.

2.3) Módulo de processos

O módulo de processos de forma geral mantém informações específicas do processo, além de possuir capacidade de agrupar os processos em quatro níveis de prioridades definidas no módulo de filas.. Toda a gerência de processos é feita por meio de chamadas do sistema realizando operações como criação, e suspensão.

2.4) Módulo de memória

O módulo de Memória tem basicamente 2 responsabilidades

a) Receber um processo e busca um espaço na memória para que seja alocado um espaço para tal processo na memória. Isso foi feito de forma que, ao identificar todos os espaços livres na memória, o módulo ordena em ordem crescente de tamanho de forma que o processo seja sempre alocado na menor área contígua de memória que ele caiba (best-fit), reduzindo assim um pouco da fragmentação externa.
b) Receber um processo e um endereço na memória e liberar esse espaço desalocando esse processo

2.5) Módulo de recurso

O módulo de recurso, na aplicação, representa o gerenciamento de E/S com um pequeno subgrupo de recursos, que são 1 *scanner*, 2 impressoras, 1 modem e 2 dispositivos SATA. Por se tratar de um Pseudo-SO, todas as operações envolvendo esses recursos são simuladas, com seu tempo de uso representado por um *sleep* randômico (entre 1 e 5 segundos). Todos os recursos, quando solicitados, necessitam dos parâmetros *processID e priority* para iniciar o seu uso.

O controle de acesso aos recursos é feito com semáforos, cuja capacidade é igual à quantidade de dispositivos disponíveis para uso. Além disso, para garantir o compartilhamento dos recursos entre múltiplos processos sem ônus para o nosso Pseudo-SO, foi criado um buffer de processos bloqueados, que garante que cada recurso será alocado para um processo por vez.

2.6) Módulo de PseudoOS

O módulo de PseudoOS tem basicamente três responsabilidades:

a) Ler os arquivos de entrada e passar cada informação para seus respectivos módulos.
b) O método “create_processes” será executado em uma thread separada que é responsável por criar o processo no momento certo, solicitar à memória o espaço para alocação da mesma e adicionar o processo na fila de processos;

c) Escalonamento de processos.

O escalonador de processos é executado em uma thread separada. No início da sua execução, o escalonador pega todos os processos que estavam bloqueados no módulo de recurso e que estão aguardando retornar à fila de pronto e os insere novamente na fila de prontos. Em seguida, é chamado o método de aging do módulo de fila e o PseudoOS solicita ao módulo de fila qual é o próximo método a ser executado, se não houver, ele reinicia todas as etapas acima descritas. Caso haja, ele solicita a instância do processo para o módulo de processos e manda executar o método “run” do processo por um tempo “x”, sendo “x” o quantum ou se for processo em tempo real, executa todas as suas “n” instruções.

No fim da execução do “run”, o processo retorna um código, que poderá representar três estados:

a) Finalizado: Então o PseudoOS solicita ao módulo de processos para deletar a instância do processo e a pede ao módulo de memória para desalocar o processo da memória;
b) O processo não finalizou e também não solicitou um recurso de E/S, então ele deve retornar à fila para ser escalonado;
c) Solicitou um recurso, então o processo não volta à fila de prontos e inicia uma thread como se a CPU tivesse solicitando aos respectivos controladores de E/S um recurso para o processo. Quando o dispositivos de E/S terminar ele vai colocar numa fila do módulo de recurso indicando ao Sistema Operacional que o processo deve voltar à fila de pronto.

3) Dificuldades

A maior dificuldade foi na divisão do trabalho de forma que cada integrante pudesse desenvolver de modo independente. Para isso, fizemos várias reuniões para definir todas as interfaces, entradas e saídas esperadas, e definir todas as responsabilidades de cada módulo e decompor a responsabilidade de cada módulo em métodos. E então, foi possível iniciar o desenvolvimento independente. A vantagem foi que definindo bem todas as responsabilidades e interfaces, cada um consegue implementar sua própria solução independentemente.

Outra dificuldade encontrada foi na questão de variáveis compartilhadas, pois sendo cada módulo independente e quem executa é apenas o PseudoOS, este módulo que dirá se haverá compartilhamento de variáveis é por consequência deve-se implementar o controle de acesso. A dificuldade foi em determinar onde seriam implementados os Locks. Por fim, decidimos delegar isso a cada método de cada módulo, de tal forma que o PseudoOS não tenha que se preocupar com exclusão mútua e possa continuar fazendo as chamadas de métodos normalmente.

4) Funções

A divisão do trabalho foi a seguinte. O Pedro ficou responsável pelo módulo de memória e fila, o Dupin com os módulos de arquivos e recursos, O Higor o módulo de processos e a main e o Alexandre o Pseudo OS.