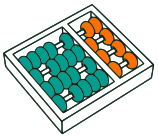

MC504— Sistemas Operacionais
INSTITUTO DE COMPUTAÇÃO — UNICAMP
1º Semestre 2018

UNICAMP

Escalonamento em Minix

Projeto 3

Resumo

Este projeto consiste em construir um administrador de escalonadores para processos a nível de usuário em Minix. O objetivo do projeto é familiariza-lo com o desenvolvimento de algoritmos e políticas de agendamento através de sua implementação no kernel do sistema operacional.

1 Escalonamento em Minix

O escalonamento em Minix se realiza através do servidor *scheduler* (`sched`) a nível de usuário. O servidor de administração de processos (`pm`) se encarrega de atribuir o escalonador a cada um dos processos que se geram. O servidor `sched` se encarrega de bloquear os mensagens do `pm` ou mensagens que estão fora do período de tempo de cada processo. Ver o reporte técnico de Swift e Tanenbaun [1, Secção 4] para os detalhes do escalonamento e sua interação com o administrador de processos.

O escalonador permite ter domínios de escalonamento onde o `pm` lhe entrega o controle a um escalonador que toma o role de administrador e permite gerar políticas distintas para os processos. De tal maneira, podemos ter múltiplos escalonadores que trabalhem em conjunto. O objetivo de este projeto é a implementação de dito administrador e a construção de dois escalonadores simples.

2 Administrador de escalonadores

A primeira tarefa para a resolução da escalonamento múltiplo é a construção de um escalonador que delegue os distintos tipos de escalonamento segundo as seguintes regras.

Você deve ter uma política *round robin* (RR) para os processos normais que chegam ao escalonador que implementou. Quer dizer, os processos pais u originais. Adicionalmente, sua administrador deve distinguir os processos filhos dos pais, e atribuir-lhes uma política distinta. Para isso, quando um processo (escalonado por este administrador) realize um `fork`, o administrador deverá atribuir-lhe aos filhos uma política distinta. Neste caso, deve implementar um `first come first serve` (FCFS). Consequentemente, os primeiros `fork` se executarão sequencialmente.

3 Implementação

Para a implementação de este projeto você tem a liberdade de implementá-lo segundo o considere conveniente. Deve justificar suas decisões de projeto (*design*) e estruturas utilizadas no relatório.

Note que deve prestar principal atenção a explicar por que cria sua solução de tal jeito, e de explicar a funcionalidade e resultados esperados. Explique pelo menos:

- Onde cria sua solução (gera um novo server, é uma chamada que modifica um server existente, etc.) e por o que a cria de essa maneira (quer dizer, se é uma chamada a sistema, um serviço, etc.).
- As estruturas de dados que gera para poder dar solução a seu problema.
- Justifique a extensibilidade e usabilidade do mesmo.

4 Provas de sua solução

Para validar a construção de seus escalonadores e das funcionalidades implementadas, deve construir um conjunto de provas unitárias (usando o Continuous Integration de Gitlab) que avaliem as funcionalidades que implementarão. Quer dizer, ao menos se espera que gere uma prova unitária por cada função e requerimento estipulado nas secções anteriores.

5 Instruções

5.1 Trabalho a realizar

1. Este projeto sera desenvolvido em grupos de três pessoas.

2. Note que este projeto não tem restrições ou guias como os anteriores. De tal maneira que se espera que gere um projeto e escolha as opções que considere pertinentes baseando-se na lógica e em métodos de engenharia.

3. Precisaram mais informação que as referencias provistas neste enunciado. Busquem informação enquanto resolvem os problemas. E mais importante, documentem-na para explicá-la em seu relatório.

4. Devem utilizar uma nova branch no seu repositório `git` para manter e versionar seu código.

5. Se revisará o avance e contribuição de cada um dos participantes do grupo através de **reportes** (autoavaliação) que relacionem o trabalho feito com o código no repositório.

5.2 Entregáveis

Devem entregar no Classroom

- `relatorio.tex`: o código fonte para gerar seu relatório, e
- `relatorio.pdf`: o arquivo de seu relatório.

Caso o código fonte do seu relatório tenha vários arquivos (os arquivos criados pelo LaTeX não devem ser inclusos) devem empacotar eles e subir um arquivo só. **Deve nomear o `pdf` usando os RAs dos integrantes do grupo.**

5.3 Data de entrega

A data de entrega é no dia 14 de junho. Deverão subir os arquivos no Classroom dentro da tarefa do projeto. Não precisam duplicar as submissões. Uma pessoa por grupo deverá submeter o projeto.

Se recomenda submeter os trabalhos com antecedência e não esperar aos últimos minutos para a submissão. Não se considerarão correios nem commits enviados depois da data limite (independentemente do motivo do atraso).

As avaliações de avanço **individual** serão enviadas com um formulário nas datas **22 de maio** e **05 de junho**.

5.4 Relatório

Devem preparar um relatório de **máximo 4 páginas** de conteúdo (quer dizer, que o limite não inclui figuras e referencias) utilizando `IEEEtran.cls`.

O relatório deve conter (pelo menos)

- Resumo
- Introdução: contextualização de escalonamento em Minix, sua estrutura, e do serviço existente (en específico, pelo menos deve explicar `sched` e seu funcionamento).
- No corpo do trabalho devem desenvolver o projeto (*design*) de sua solução em detalhe (de tal forma que outro desenvolvedor possa duplicar seu trabalho). Este detalhe deve incluir descrições de estruturas de dados, algoritmos utilizados e equações ou fórmulas (no triviais), descrição de cada função relevante (incluindo seu propósito, entradas, saídas, e supostos sobre estes).

  Note que dada a liberdade entregada no projeto, a justificação de design é de máxima importância para entender seu trem de pensamento e a solução planteada.

  Deve de explicar o projeto e fundamentação de pelo menos:

  - Definição e criação do administrador dos escalonadores
  - Definição e criação dos escalonadores: FCFS e RR
  - Provas unitárias (explicação sobre as considerações da construção das provas unitárias e como estas validam sua proposta)
- Conclusão (explicar os pontos principais do trabalho)

Se se detecta plagio os envolvidos terão zero na disciplina, e se dará aviso às autoridades correspondentes para que se tomem as sanciones do caso.

6 Dicas

- Revisem a documentação oficial de Minix para poder resolver problemas. A informação deste documento é um ponto de partida e não uma guia de tudo o que tem que ser feito. Revise em especial as referencias de este documento [1] para entender o que faz o escalonador em Minix.
- Estabeleçam metas iniciais, e etapas para avançar no projeto. Por exemplo, instalar Minix, logo comunicar-se e instalar programas através de `ssh`, etc. A tarefa pode demorar mais se não são organizados no uso de seu tempo, e se ficam refazendo as mesmas coisas uma e outra vez não estão avançando.
- Criem seu projeto de maneira incremental. E tenham uma maneira de avaliar a funcionalidade de seu código (o script de teste) que permita a vocês saber instantaneamente se seu código não está funcionado.

- Avalie seu sistema constantemente. Escreva um conjunto de provas que possa executar recorrentemente (através da integração continua) para verificar que seu sistema funciona. Avalie cenários exitosos assim como também aqueles que contenham erros. Por exemplo, você deve ter certeza que uma função que não recebe inteiros da erro quando recebe um número inteiro.
- **Não comece o projeto dias antes da submissão**. Planejem sua entrega e façam avanços oportunamente.
- Neste projeto será avaliado o uso do repositório e as contribuições individuais. **Se você não tem commits no repositório será penalizado por não poder demostrar uma contribuição no grupo.**
- Se assume que você está familiarizado com o uso de `Makefiles` e técnicas de depuração (introduzidas em disciplinas anteriores). De não ser o caso, deverá aprender estas ferramentas também. O projeto não requere demasiadas linhas de código, mas requere que entenda Minix e como utilizar as chamadas básicas do sistema.

7 Avaliação

O projeto contempla a avaliação do relatório (escrita, estruturação, redação, etc.) e as tarefas feitas (prática) de maneira conjunta. Se avaliará o relatório entregue, assim como o trabalho feito no repositório (através do histórico no repositório). O detalhe da avaliação esta dado na Tabela 1. A parte prática também será avaliada a través de suas provas unitárias.

A rubrica será utilizada segundo os quatro critérios definidos por cada dimensão que explicitam os rangos porcentuais que podem ser obtidos segundo o desempeno observado. Excepto na dimensão "escrita técnica" que será avaliada por um único critério.

Referências

[1] B. P. Swift and A. Tanenbaum, "Individual programming assignment user mode scheduling in minix 3," 2010.

Tabela 1. Rúbrica para a avaliação do projeto.

Dimensión	**Malo** 0%	**Bom** 1%–50%	**Muy Bom** 51%–85%	**Excelente** 86%–100%	**Punteo**
Informe					
Resumo	Reproduz o enunciado do projeto entregue, ou uma descrição pobre das tarefas realizadas sem explicar pontos principais do trabalho.	Explica a motivação, o problema, e, *quando muito um* dos seguintes, a solução encontrada, os resultados principais, ou a conclusão principal encontrada.	Explica a motivação, o problema, e, *quando muito dois* dos seguintes, a solução encontrada, os resultados principais, ou a conclusão principal encontrada.	Explica a motivação, o problema, a solução encontrada, os resultados principais, e a conclusão principal encontrada.	2
Introdução	Reproduz o enunciado do projeto entregue, ou uma descrição pobre das tarefas realizadas sem explicar as partes do sistema utilizadas, e não agrega sua visão do problema a resolver.	Explica as partes do sistema utilizadas, e o problema a resolver (embora não entra em detalhe); mas não explica como elas se relacionam para resolvê-lo.	Explica as partes do sistema utilizadas, e como se relacionam para resolver o problema; mas este não é explicado em detalhe.	Explica a importância das partes do sistema ou enuncia o problema a resolver formalmente, detalha as partes do sistema utilizadas, e como se relacionam para resolver o problema.	3
Administrador	Não explica a estrutura de arquivos, nem o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração dos escalonadores em Minix. Não explica o por que das ideias implementadas, e não justifica as decisões de projeto.	Explica, para que outro programador possa desenvolver usando o código fonte e com informação adicional, a estrutura de arquivos e o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração dos escalonadores em Minix. Explica o por que da implementação e justifica as decisões de projeto, em *alguns (menos do* 60%*)* dos casos.	Explica, para que outro programador possa desenvolver usando o código fonte, a estrutura de arquivos e o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração dos escalonadores em Minix. Explica o por que da implementação e justifica as decisões de projeto, *na maioria (mais do* 60% *inclusive)* dos casos.	Explica, para que outro programador possa desenvolver sem informação adicional, a estrutura de arquivos e o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração os escalonadores em Minix. Explica *detalhadamente (mais do* 85% *inclusive)* o por que da implementação e *justifica todas* as decisões de projeto.	15
Escalonador	Não explica a estrutura de arquivos, nem o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração das políticas de escalonamento em Minix. Não explica o por que das ideias implementadas, e não justifica as decisões de projeto.	Explica, para que outro programador possa desenvolver usando o código fonte e com informação adicional, a estrutura de arquivos e o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração das políticas de escalonamento em Minix. Explica o por que da implementação e justifica as decisões de projeto, em *alguns (menos do* 60%*)* dos casos.	Explica, para que outro programador possa desenvolver usando o código fonte, a estrutura de arquivos e o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração as políticas de escalonamento em Minix. Explica o por que da implementação e justifica as decisões de projeto, em *la maioria (mais do* 60% *inclusive)* dos casos.	Explica, para que outro programador possa desenvolver sem informação adicional, a estrutura de arquivos e o projeto (entradas, saídas, e lógica de funcionamento) das funções encarregadas da criação e administração as políticas de escalonamento em Minix. Explica *detalhadamente (mais do* 85% *inclusive)* o por que da implementação e *justifica todas* as decisões de projeto.	12

Dimensión	**Malo** 0%	**Bom** 1%–50%	**Muy Bom** 51%–85%	**Excelente** 86%–100%	**Punteo**
Provas unitárias	Não explica a definição de suas provas unitárias para as funcionalidades solicitadas, nem justifica como estas validam as funcionalidades.	Explica, *pelo menos com pseudocódigo*, a definição de seus provas unitárias para as funcionalidades solicitadas, e justifica como estas validam *menos do* 60% das funcionalidades.	Explica *(inclui pseudocódigo e sua explicação embora não necessariamente o justifica)* a definição de suas provas unitárias para as funcionalidades solicitadas, e justifica como estas validam *mais do* 60% *inclusive* das funcionalidades.	Explica *detalhadamente (inclui pseudocódigo e justificações)* a definição de seus provas unitárias para as funcionalidades solicitadas, e justifica como estas validam *mais do* 85% *inclusive* das funcionalidades.	5
Conclusões	Apresenta argumentos não relacionados com o trabalho desenvolvido, *apreciações pessoais* (por exemplo, que se aprendeu muito no projeto, ou que foi difícil), mas não se desenvolvem argumentos que fundamentem o realizado.	Explica as implicações chave ou resultados principais, mas não entra em detalhe sobre o argumento principal ou os trabalhos futuros.	Explica, ao menos dois das seguintes, o argumento principal, as implicações chave, ou os resultados principais, e explica com pouco detalhe os argumentos ou trabalhos futuros.	Explica o argumento principal, as implicações chave, resultados principais, repercussões e detalhe dos argumentos apresentados, e o trabalho futuro.	3
Escrita técnica	Organiza a explicação do desenvolvimento no documento de maneira coerente e clara. Explica o processo de desenvolvimento de maneira coerente e detalhada. Escreve sem faltas de ortografia e redação. (Esta dimensão se analisará num critério único, quer dizer, se ponderará pelo 100%.)				10
Prática					
Administrador	Executa corretamente *até o* 25% das provas relacionadas à delegação de pais e filhos em seus respectivas políticas de escalonamento.	Executa corretamente *mais do* 25% das provas relacionadas à delegação de pais e filhos em seus respectivas políticas de escalonamento.	Executa corretamente *mais do* 70% das provas relacionadas à delegação de pais e filhos em seus respectivas políticas de escalonamento.	Executa corretamente *mais do* 85% das provas relacionadas à delegação de pais e filhos em seus respectivas políticas de escalonamento.	10
FCFS	Executa corretamente *até o* 25% das provas relacionadas à política de escalonamento.	Executa corretamente *mais do* 25% das provas relacionadas à política de escalonamento.	Executa corretamente *mais do* 70% das provas relacionadas à política de escalonamento.	Executa corretamente *mais do* 85% das provas relacionadas à política de escalonamento.	6
RR	Executa corretamente *até o* 25% das provas relacionadas à política de escalonamento.	Executa corretamente *mais do* 25% das provas relacionadas à política de escalonamento.	Executa corretamente *mais do* 70% das provas relacionadas à política de escalonamento.	Executa corretamente *mais do* 85% das provas relacionadas à política de escalonamento.	6
Avances (individual)	Se avaliará a contribuição individual ao desenvolvimento de cada componente através dos `commit` individuais que se encontram no repositório oficial do curso oportunamente e distribuídos a o largo do tempo (nas entregas estipuladas). Quer dizer, a ponderação variará segundo sua contribuição ao projeto segundo os `commit` do repositório nas datas antes mencionadas.				15
Provas unitárias	Não apresenta provas próprias, que se executem, para as funcionalidades solicitadas e sua aplicação.	Apresenta provas, que se executem, para *até o* 50% das funcionalidades solicitadas e sua aplicação.	Apresenta provas, que se executem, para *até o* 80% das funcionalidades solicitadas e sua aplicação.	Apresenta provas, que se executem, para *mais do* 80% *inclusive* das funcionalidades solicitadas e sua aplicação.	13
				Total	**100**