EP1

Renê Cardozo - 9797315
Verônica Stocco - 6828626
rene.cardozo@usp.br
veronica.stocco@usp.br

Instituto de Matemática e Estatística
Universidade de São Paulo

BCCSH

O executável bccsh é composto de dois módulos: parser e bccsh. Além disso é utilizada a biblioteca readline, a qual tem seu caminho encontrado pela ferramenta pkgconfig.

Parser

- O parser é responsável por receber a linha de caracteres retornada pela função readline da biblioteca readline e separar cada palavra em um vetor de argumentos, os quais serão reconhecidos pelo bccsh posteriormente e determinarão qual comando deverá ser executado.
- Este parser foi adotado pensando em possibilitar a execução de qualquer binário pelo shell, bem como unificar o processo de execução de comandos internos e outros programas do sistema.
- Há uma limitação de 10 strings para cada linha lida pelo parser.

Shell

- O shell propriamente dito é implementado no arquivo bccsh.c, o qual realiza a leitura de inputs pelo usuário utilizando a função readline, sendo lida uma linha por vez.
- Após o armazenamento da linha em um buffer, caso a linha não seja vazia, esta será armazenada no histórico através da função add_history da biblioteca readline, podendo ser acessada, por padrão, utilizando as setas do teclado.

Comandos

Uma vez salvo no histórico, a linha será utilizada pelo parser para criar um vetor com as palavras digitadas. As quais utilizarão a função cmd para determinar qual comando será executado. Podendo este ser:

- mkdir <diretório>
- kill -<sinal> <PID>
- ln -s <arquivo> <link>
- Qualquer binário do sistema iniciado por "/" ou ".", com no máximo 9 argumentos.

Escalonador de Processos

O simulador de escalonamento de processos é definido no arquivo ep1.c, no qual estão definidas as rotinas de escalonamento com os algoritmos First-Come, First-Served (FCFS) e Shortest Remaining Time Next (SRTN).

Round-Robin

O algoritmo Round-Robin não foi implementado por falta de tempo.

Neste arquivo é utilizada a função read_file para ler os arquivos da pasta entrada, na qual são armazenados os arquivos criados pelo gerador de processos, os quais são organizados em um array de structs onde é armazenado o nome, t0, dt e deadline de cada processo.

Algoritmos de Escalonamento

A partir do primeiro parâmetro fornecido para o executável ./ep1 temos a escolha do algoritmo que será utilizado na simulação, sendo 1 para FCFS e 2 para SRTN.
A implementação dos algoritmos foi feita da seguinte forma:

- FCFS: Executa as threads por diretamente do array de processos montado na leitura, uma vez que supõe-se que os processos estejam ordenados por ordem de chegada.
- SRTN: Reordena o array de processos com base na soma dos atributos t0 e dt, assim, priorizam-se os processos com menor dt que chegam antes.

Os algoritmos armazenam a saída no arquivo especificado pelo terceiro parâmetro do executável ./ep1, no qual é escrito o nome do processo o tempo atual do simulador e o tempo do simulador menos o tempo inicial do processo.

Gerador de Processos

A criação de processos para uso pelo simulador é feito pelo arquivo gerador.c dentro da pasta gerador_de_processos. Este programa é executado da forma "./gerador <número-de-processos [semente]", sendo a semente opcional e por padrão igual a 0.

- t0 inicia-se com 1 e é incrementado em uminteiro aleatório entre 0 e 3.
- dt é gerado como um inteiro aleatório entre 2 e 5.
- a deadline do processo é gerada por t0 + dt + x, sendo x um número inteiro aleatório entre 0 e 15.

Testes

Os testes foram realizados a partir da geração de três arquivos de processos, com 5, 50 e 500 processos.
A execução de cada teste foi realizada para os dois algoritmos implementados e executadas trinta vezes para cada arquivo de 5, 50 e 500 processos.
A execução usou como base:

- ./bccsh
- ./ep1 <1 ou 2> entrada/<arquivo-de-entrada> saida/<arquivo-de-saida> d