Sistemas Operativos
Grupo 34

Alexandre Rodrigues (A81451) Pedro Ferreira (A81403)

1 de Junho de 2018

Resumo

Neste trabalho foi feito um interpretador de notebooks.

Conteúdo

1 Estratégia de Desenvolvimento

O nosso pensamento geral neste trabalho foi:

1. Passar o notebook para memoria[1]
2. Processar o conteúdo na memoria[2]
3. Converter de novo para notebook, agora com outputs

[1] Neste documento refere-se a isto como parsing
[2] Neste documento refere-se a isto como processing

1.1 Estruturas de Dados

Na produção deste trabalho decidimos que seria melhor guardar o ficheiro notebook em memória enquanto o processávamos. Isto pois, é mais fácil de manipular os parâmetros quando se encontram numa estrutura em vez de ficheiro. Facilita o acesso ao output de comandos anteriores. Achamos que o tempo de parsing do ficheiro inteiro é compensado no tempo de processamento, especialmente quando o notebook envolve vários comandos dependentes do output de outros.

Logo criou-se as duas seguintes estruturas: uma lista de comandos e o comando em si. O comando possui os parâmetros(inclusive o comando), o output que será preenchido após processamento, um int de referência que indica quão atrás se encontra o output que irá ser usado como input e um int que indica se faz parte de uma sequencia.

A lista utiliza um GPointer Array e ambas as estruturas garantem encapsulamento.

1.2 Parsing do Notebook

Para fazer parsing do notebook é utilizada uma função que recebe o array de argumentos, a quantidade de argumentos e também uma estrutura LIST para onde serão transferidos os dados do ficheiro. Esta função começa por abrir o ficheiro pretendido com um fopen e após verificar se tudo correu bem, utiliza um fgets para ler o ficheiro linha a linha.

Visto que os comandos têm um formato fixo é possível utilizar o sscanf em cada linha procurando por três formatos diferentes:

- A linha começa por "$ ", logo basta adicionar o comando à lista com ref = 0;
- Começa por "$| ",adiciona-se à lista com ref = 1;
- Ou por "$n| ", que equivale a adicionar com ref = n;

Numa sequencia, o primeiro elemento pode ser um dos três acima, enquanto que os restante sao considerados ref 1, inseridos na lista individualmente.

Quando se chega ao final do ficheiro o ciclo acaba e o ficheiro é fechado.

Após o processamento é necessário passar a LIST de volta a notebook, para o qual é utilizada uma função que recebe os mesmos argumentos que a anterior e que abre o mesmo ficheiro, mas desta vez remove todo o seu conteúdo.

É depois criado um ciclo que percorre a estrutura LIST e verifica se os comandos pertenciam a uma linha de comandos seguidos ou não, através do inteiro sline passado para a LIST no momento de parsing da função anterior. Caso negativo basta verificar se a referência do comando é 0 ("$ "), 1 ("$| ") ou maior ("$n| ") e imprime-se para o ficheiro o comando seguido do seu output.

Caso contrário é colocado um ciclo adicional para imprimir todos os comandos por ordem e no fim o seu output calculado.

1.3 Processamento

Para processar a estrutura de memória seguem-se os seguintes passos:

- Abrir um ficheiro temporário *tmp* para guardar o input dos comandos
- Abrir um pipe anónimo, para receber o output
- Iniciar o ciclo no qual corremos os comandos da seguinte forma:
  - Verifica-se se o comando tem input adicional(ref > 0), se sim é posto no tmp.
  - Faz-se um fork. No filho redirecionámos o stdin para tmp e stdout para o pipe. O pai, espera que o filho acabe.
  - O conteúdo do pipe é lido para o output do comando e avança-se para o comando seguinte.

1.4 Prevenção de erros

O programa está preparado para erros de leitura, forking, etc. Implementa também um signal_handler que apaga o ficheiro temporário no caso de terminação.

2 Opções Avançadas

Como referido anteriormente implementou-se a funcionalidade de ler uma sequência de comandos. Esta é tratada inteiramente pelo parser logo que, pela altura que chega a memória é indistinguível de várias linhas separadas a usar o resultado anterior. O parser utiliza o int sline para determinar que argumentos fazem parte da mesma linha durante a reconstrução.