Trabalho de Sistemas Operativos
Processamento de um notebook

Sérgio Oliveira
a62134

Pedro Dias
a63389

21 de Maio de 2018

Conteúdo

1 Introdução **2**

2 Desenvolvimento **3**
2.1 Funcionalidades básicas 3
2.1.1 Execução de programas 3
2.2 Re-processamento de um notebook 3
2.3 Detecção de erros e interrupção da execução 3

3 Funcionalidades avançadas **4**
3.1 Acesso a resultados de comandos anteriores arbitrários 4
3.2 Execução de conjuntos de comandos 4

4 Conclusão **5**

Capítulo 1

Introdução

Este relato´rio tem como objetivo demonstrar o desenvolvimento de um programa que faz o processamento de ficheiros no formato designado pelo trabalho pra´tico.

Um ficheiro de formato notebook tem a extensa˜o '.nb'. Este pode conter linhas de co´digo para serem interpretadas pela shell do sistema, resultados de execuça˜o das respetivas linhas de co´digo bem como outro tipo de conteu´do tal como documentaça˜o relevante.

Relacionado com os temas mencionados, serão apresentados ao longo do relatório fragmentos de co´digo para que haja uma melhor compreensa˜o do assunto em questão.

O relato´rio encontra-se dividido em va´rias secço˜es que correspondem aos pontos do enunciado do trabalho.

Capítulo 2

Desenvolvimento

2.1 Funcionalidades básicas

2.1.1 Execução de programas

Neste ponto, o programa deve interpretar as linhas começadas por $ como comandos (programa e argumentos), e se começadas por $| o comando deve ter como stdin o resultado do comando anterior. O resultado produzido é então colocado no ficheiro de input e delimitado por "»»" e "«««". O exemplo escolhido foi o fornecido no enunciado do trabalho.

Para a leitura do ficheiro comandos .nb e de todos os ficheiros passados como argumento, foi utilizado um ciclo *while* que faz a leitura linha a linha até ao fim do ficheiro. A cada iteração do ciclo, é enviado para o ficheiro "out.txt" (previamente criado) a linha lida até encontrar o primeiro sinal de $. Após encontrar este primeiro comando, é feito o parsing dessa linha para separar o sinal de $ e o comando. Para isto é utilizado a função auxiliar "trim".

Posteriormente é criado um novo ficheiro, "result1.txt" (colocado no diretório "tmp" criado no início do programa), que é utilizado como stdout da execução da system call "execl". É criado um fork para a execução do "execl" anteriormente descrito e é feita a espera do processo "filho" pelo processo "pai" para que depois seja transferido todo o conteúdo do ficheiro "result1.txt" delimitado por "»»" e "«««" para o ficheiro "out.txt".

A lógica de execução dos comandos seguidos por $| é a mesma seguida por $ mas com a diferença que esta execução tem como stdin o resultado anterior, ou seja, neste caso a execução do comando "sort" vai ter como stdin o ficheiro "result1.txt" e como stdout o ficheiro "result2.txt".

2.2 Re-processamento de um notebook

O re-processamento de um notebook é processado através da função auxiliar "re$_{p}$*rocessamento*ı.*Esta funotem*

2.3 Detecção de erros e interrupção da execução

A deteção de erros é verificada a cada execução de comandos. Após a criação do fork o processo pai fica à espera do processo filho e, se neste ocorrer algum erro na execução do comando, é executada a linha de código "exit(-1)" e o processo pai apanha o sinal de erro, terminando o processamento do ficheiro de input.

A detecção da interrupção da execução (sinal normalmente relacionado com a combinaça˜o de boto˜es Control+C) é feita através do envio do sinal SIGINT para o programa. Nesse caso, a varia´vel global running ira´ determinar o estado de execuça˜o do nosso programa. Na início da função "main" do programa é colocado a linha "signal(SIGINT, handler);" que detecta o sinal SIGINT e invoca a função "handler" que altera o estado da variável "running" e termina o programa sem alterar o estado do ficheiro de input.

Capítulo 3

Funcionalidades avançadas

3.1 Acesso a resultados de comandos anteriores arbitrários

O execução dos comandos seguidos por "$N|" contém a mesma lógica de execução dos sinais $ e $| com a particularidade de que o stdin do novo comando vai ser o ficheiro "resultN.txt" em que N é o número entre "$" e "|" . Para reconhecer o sinal "$N|" utiliza-se a expressão regular " $[1-9][0-9]*| " e a biblioteca "regex.h" para se fazer o parsing da linha.

3.2 Execução de conjuntos de comandos

A execução de conjuntos de programas foi testado com o exemplo abaixo e o "execl" utilizado no programa executa o comando com pipes como pretendido.

Capítulo 4

Conclusão

Apo´s 2 semanas de desenvolvimento, ha´ que realçar alguns pontos importantes, como a dificuldade de implementar funçoes de baixo ni´vel (implementaçao de pipes com nome, correta comunicaçao entre forks, entre outros) seguindo assim outra direcção para a realização do trabalho pratico.

A utilização de ficheiros temporários como stdin e stdout foi discutida por nós e vista como a solução para completar todos os pontos colocados no enunciado do trabalho.

A única excepção foi no ponto 2.2 , em que o grupo decidiu não avançar para a execução em background dos ficheiros. O grupo de trabalho sentiu dificuldades em conciliar a realização deste projecto com os outros projectos do curso e ainda com o trabalho de dia de cada um , mas mesmo com estes obstaculos ficamos a compreender mais sobre as system calls de baixo ni´vel e em que contexto usa´-las, para termos um maior controlo de processos e ficheiros. Com o desenvolvimento destes projetos, havera´ tambe´m sempre a expansao do nosso conhecimento e interesse em sistemas UNIX e UNIX-like.