Projeto de Sistemas Operativos (MiEI)
2017/2018

Alexandre Mendonça Pinho (a82441)
Joel Filipe Esteves Gama (a82202)
Tiago Martins Pinheiro (a82491)

5 de Junho de 2018

Conteúdo

1 Introdução **2**

2 Descrição do projeto **3**

3 Exemplo **4**

4 Conclusão **8**

Capítulo 1

Introdução

Neste projeto é-nos proposto construir um sistema para processamento de notebooks, que misturam fragmentos de código, resultados da execução, e documentação. Estes notebooks tratam-se de ficheiros de texto que são modificados de modo a incluir os resultados da execução dos comandos neles embebidos.

Capítulo 2

Descrição do projeto

Para resolver o problema que nos foi apresentado começamos por fazer o parsing do ficheiro. O parsing excluí o output dos comandos (se presente) da escrita final e constroi o grafo de execução, cujos nodos correspondem a comandos a executar e cujas arestas são os redicionamentos dos outputs dos comandos. Se um nodo A do grafo tem uma aresta vinda de outro nodo B, isto significa que o output de B será redirecionado como o input de A. Como um nodo apenas pode conter uma origem do seu input, o grafo formado será um conjunto de árvores. A raiz de cada uma destas árvores recebe o seu output, por ordem de execução como o ficheiro de input sugere, do standard input. Em cada uma destas árvores, os processos são executados em paralelo, mas como em cada instante não é desejável estarem em execução dois processos que leem do standard input, o programa espera que o anterior dos dois termine ante de iniciar o seguinte.

Para além de alimentar os outros processos com o seu input se necessário, o output dos comando é também escrito num FIFO reservado a ser lido pelo programa principal, para posteriormente ser escrito de volta para o notebook.

Aquando da execução dos comandos, o seu output é primeiro redirecionado para um outro programa, um distribuidor, através de um pipe anónimo. O programa distribuidor recebe como argumentos os nomes dos ficheiros para os quais deve escrever o que lê no seu input. Serve assim para duplicar e distribuir o output de um processo para vários FIFOs.

Depois da execução de todos os comandos, os seus outputs são incluidos no string (representado como uma lista ligada de blocos contendo cada um um buffer, para mais fácil introdução de novo conteúdo) que será escrito para ficheiro. É apenas no final que este string é escrito para ficheiro, permitindo ao utilizador cancelar o processamento do notebook a meio do programa sem alterar o ficheiro de input.

Capítulo 3

Exemplo

Ficheiro de input exemplo:

```
Listagem da diretoria atual
$ ls

O número de ficheiros na listagem
$ | wc -l

O nome do primeiro elemento
$ 2 | head -1

Aceitar input do utilizador
$ cat
```

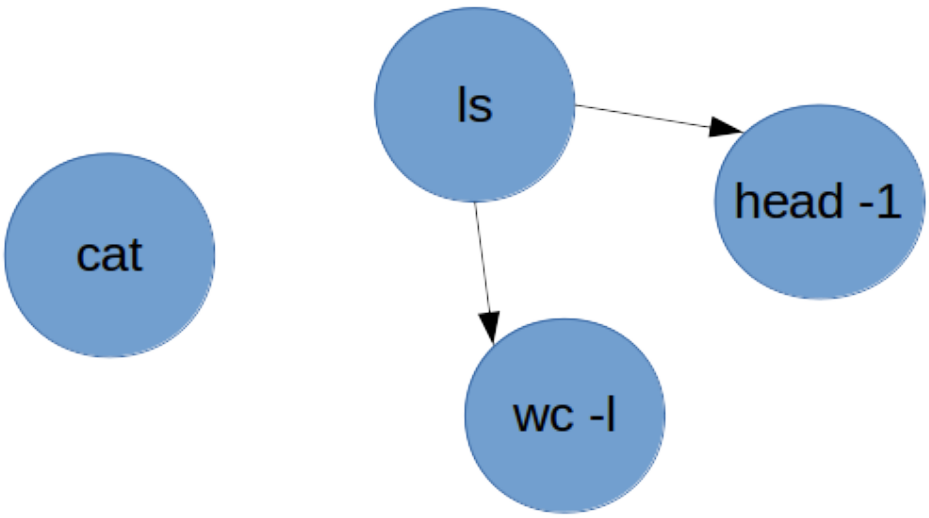

Figura 3.1: Grafo de execução

A figura 3.1 mostra a representação visual do grafo de execução contruido pelo parsing do ficheiro de input. O output do comando `ls` é redirecionado para o comando `wc -l`, que conta o número de linha da listagem da diretoria, e para o comando `head -1`, que imprime o nome do primeiro elemento da listagem. O grafo contém tambem o comando `cat`, que nos permite introduzir algum texto.

A figura 3.2 representa esquematicamente como o input e output dos processos são tratados. Para cada processo, é criado um FIFO em que vai ser escrito o seu output. Se esse processo aceitar o seu input de um outro processo aqui representado, como é o caso dos comandos `wc -l` e `head -1`, é também criado um FIFO onde será escrito o output do respetivo comando. Caso contrário (nos comandos `ls` e `cat`), o seu input é lido do standard input.

O output de cada processo é redirecionado para o input do seu respetivo distribuidor através de um pipe anónimo. Este distribuidor foi inicializado com o nome do FIFO de output e, no caso dos processos cujo input é pedido por outros processos, também com os nomes dos FIFOs de input dos ditos processos.
Ficheiro de output resultante, ao introduzir pelo no teclado o string "Hello, world!\n"(seguido da tecla ctrl-d para terminar o input):

```
Listagem da diretoria atual
$ ls
>>>
```

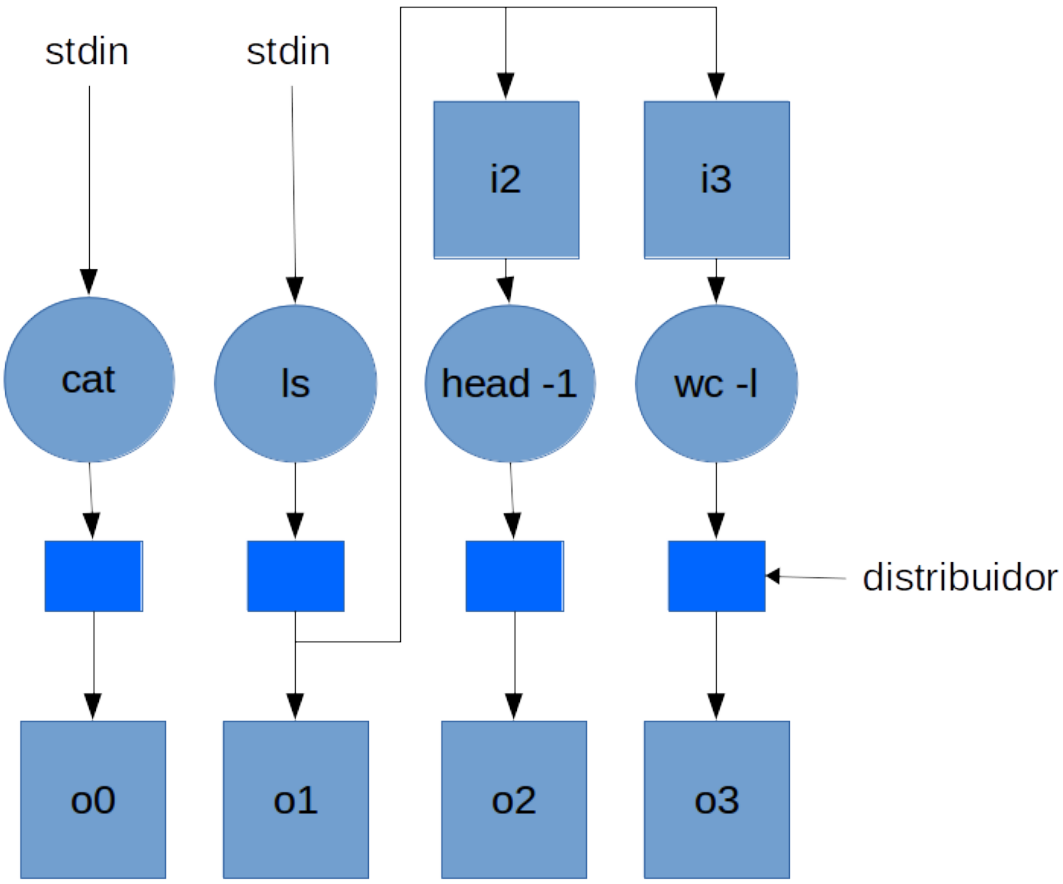

Figura 3.2: Esquema de execução

```
distribuidor
distribuidor.c
i1
i2
include
lib
main.c
Makefile
notebook
o0
o1
o2
o3
obj
README.md
relatorio
test
<<<
O número de ficheiros na listagem
$ | wc -l
>>>
17
<<<
O nome do primeiro elemento
$ 2 | head -1
>>>
distribuidor
<<<
Aceitar input do utilizador
$ cat
>>>
Hello, world!
<<<
```

Capítulo 4

Conclusão

Foi-nos prosposto, como projeto de avaliação, conceber um sistema que fosse capaz de processar *notebooks*, através da sua leitura e interpretação. A implementação foi feita na linguagem de programação *C*, utilizando os conceitos aprendidos durante as aulas práticas ao longo do semestre.

Para os métodos de comunicação entre processos, escolhemos pipes anónimos para comunicar entre os processos e os distribuidores, e *fifos* para comunicar entre os diferentes processos através dos distribuidores. Assim, a comunicação entre processos é tratada de forma simples pelos distribuidores.