Universidade do Minho

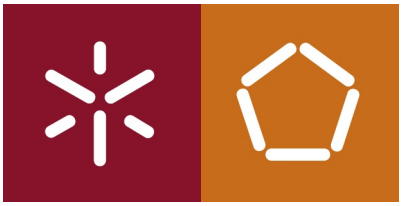
Mestrado Integrado em Engenharia Informática

Sistemas Operativos

Processamento de Notebooks

Autores:
João Gomes (A74033)
Joel Morais (A70841)
Tiago Fraga (A74092)

2 de Junho de 2018



Conteúdo

Capítulo 1

Introdução

O projeto apresentado neste relatório surge no âmbito da Unidade Curricular de Sistemas Operativos do 2º ano do Mestrado Integrado em Engenharia Informática, com o objetivo de solidificar os conhecimentos adquiridos durante o semestre, tendo como tema principal a criação de um sistema de *processamento de notebooks*.

Um *notebook* é um ficheiro de texto que depois de processado é modificado de modo a incorporar resultados da execução de código ou comandos nele embebidos.

Este processamento consiste em interpretar as linhas começadas por *$* como comandos que serão executados e o resultado produzido inserido imediatamente a seguir. E as linhas começadas por *$|* executam comandos que dependem do resultado produzido anteriormente.

Começamos este projeto por implementar funcionalidades básicas, como a **Execução de Programas**, ou seja, reconhecimento das linhas e execução dos comandos descritos nessas linhas. **Re-processamento de um notebook** garantir que o *notebook* possa ser editado e processado novamente. E por último implementamos a funcionalidade de **Deteção de erros e interrupção da execução** que consiste em abortar o processamento, devendo o *notebook* ficar inalterado.

Ao longo deste relatório será explicado o raciocínio e decisões que o grupo tomou durante a a implementação do projeto, bem como apresentada a arquitetura do sistema final.

Capítulo 2

Funcionalidades básicas

Tal como referido de forma breve na introdução começamos por implementar as funcionalidades básicas. Para que tal aconteça inicialmente, filtramos as linhas começadas por *$* de modo a podermos retirar o comando e os seus argumentos de modo a executarmos o comando pretendido. Depois de conseguirmos executar um comando sozinho, seguimos para o processamento de uma linha com *$|* de modo a executarmos um comando que dependia da execução do anterior. Para darmos este passo optamos pela utilização de de **pipes anónimos** de modo a poder enviar o output do comando anterior para o input do comando a executar, além disso usamos um conjunto de ficheiros auxiliares, um por comando lido, para guardar o output produzido pelos mesmos.

O nosso programa permite que o notebook, seja alterado, substituindo os resultados de processamento anterior pelos novos resultados. Isto acontece porque a medida que efetuamos o processamento anterior vamos guardando num ficheiro os resultados e só no fim do processamento é que voltamos a escrever no ficheiro de teste.

Como na parte de deteção de erros, quando é detetado algum, temos de manter o ficheiro original inalterado, logo achamos que esta estratégia do uso de ficheiros auxiliares era a melhor para garantir estas condições. Neste ponto utilizamos um sinal para informar o processo pai que um erro ocorreu e assim interromper a execução do comando.

```
while((linha = lerLinha(fd))!=NULL){

      if (linha[0] == '$' && linha[1] == '|'){

        write(fd_result,linha,strlen(linha));
        write(fd_result,"\n",1);

        for(token = strtok(linha+2, " "), i = 0; token; token = strtok(NULL, " "), i++)
          args[i] = token;

        args[i] = NULL;
        write(fd_result,">>>\n",4);

        pipe(p);

        if(fork()==0){
          close(p[0]);
          char* name_ant = (char*) malloc(sizeof(char)*10);
          sprintf(name_ant,"%d.txt",j-1);

          file = open(name_ant, O_RDONLY);

          dup2(p[1], 1);
          dup2(file, 0);
          execvp(args[0], args);
          kill(getppid(),SIGUSR1);
```

```
        _exit(0);
      }

      wait(&status);
      close(p[1]);

      signal(SIGUSR1,handler);
      if(controlo == 0){
        sprintf(name,"%d.txt",j);

        fd_tmp[j] = open(name,O_CREAT | O_RDONLY |O_WRONLY,S_IRWXU | S_IRWXG | S_IRWXO);
        if(fd_tmp[j]!=-1){

          while((linha = lerLinha(p[0]))){
            write(fd_result, linha, strlen(linha));
            write(fd_tmp[j], linha, strlen(linha));
            write(fd_tmp[j],"\n",1);
            write(fd_result,"\n",1);
          }

          close(p[0]);
          close(fd_tmp[j]);
          j++;
          write(fd_result,"<<<\n",4);
        }else{
          perror("NAO ABRIU O FICHEIRO");
          _exit(1);
        }
      }else{
        remove("result");
        perror("COMANDO IMPOSSIVEL DE EXECUTAR");
        _exit(1);
      }
```

Aqui evidenciado temos o inicio do ciclo *while* que vai ler o *notebook* e o primeiro *if* que trata os casos em que temos um *$* seguidos /, ou seja, um comando dependente do resultado do comando anterior.

```
else if(linha[0] == '$'){

      write(fd_result,linha,strlen(linha));
      write(fd_result,"\n",1);

      for(token = strtok(linha+1, " "), i = 0; token; token = strtok(NULL, " "), i++){
        args[i] = token;
      }

      args[i] = NULL;
      write(fd_result,">>>\n",4);

        pipe(p);


      if(fork()==0){
        close(p[0]);
        dup2(p[1], 1);
        execvp(args[0], args);
        kill(getppid(),SIGUSR1);
        _exit(0);
      }
```

```
wait(&status);
close(p[1]);
signal(SIGUSR1,handler);

if(controlo == 0){
  sprintf(name,"%d.txt",j);

  fd_tmp[j] = open(name,O_CREAT | O_RDONLY |O_WRONLY,S_IRWXU | S_IRWXG | S_IRWXO);

  if(fd_tmp[j]!=-1){

    while((linha = lerLinha(p[0]))){
      write(fd_result, linha, strlen(linha));
      write(fd_tmp[j], linha, strlen(linha));
      write(fd_tmp[j],"\n",1);
      write(fd_result,"\n",1);
    }

    close(p[0]);
    close(file);
    close(fd_tmp[j]);
    j++;
    write(fd_result,"<<<\n",4);
  }else{
    perror("NAO ABRIU O FICHEIRO");
    _exit(1);
  }
```

Ainda relativamente as funcionalidades básicas temos o caso mais simples em que a linha possuí só um *$*, ou seja, só um comando independente.

Capítulo 3

Funcionalidades Avançadas

Depois de desenvolvidas as **funcionalidades básicas**, resolvemos melhorar o nosso trabalho desenvolvendo **funcionalidades avançadas**. Para desenvolvermos a primeira funcionalidade avançada proposta fizemos uso dos ficheiros temporários, um por comando, cada ficheiro temporário possuí com nome o número, incrementado a medida que aparecem no ficheiro, mantendo assim uma ordem de ficheiro para ficheiro, assim podemos ir buscar o resultado do n-ésimo comando anterior facilmente.

```
}else if(linha[0] == '$' && linha[2] == '|'){

        write(fd_result,linha,strlen(linha));
        write(fd_result,"\n",1);

        for(token = strtok(linha+3," "), i = 0; token; token = strtok(NULL, " "), i++){
          args[i] = token;
        }

        args[i] = NULL;
        write(fd_result,">>>\n",4);

        num = linha[1] - '0';
        ant = j-num;

        pipe(p);

        if(fork()==0){
          close(p[0]);
          char* name_ant = (char*) malloc(sizeof(char)*10);
          sprintf(name_ant,"%d.txt",ant);

          file = open(name_ant, O_RDONLY);

          dup2(p[1], 1);
          dup2(file, 0);
          execvp(args[0], args);
          kill(getppid(),SIGUSR1);
          _exit(0);
        }

        wait(&status);
        close(p[1]);

        signal(SIGUSR1,handler);
        if(controlo == 0){
          sprintf(name,"%d.txt",j);
```

```
      fd_tmp[j] = open(name,O_CREAT | O_RDONLY |O_WRONLY,S_IRWXU | S_IRWXG | S_IRWXO);
      if(fd_tmp[j]!=-1){

        while((linha = lerLinha(p[0]))){
          write(fd_result, linha, strlen(linha));
          write(fd_tmp[j], linha, strlen(linha));
          write(fd_tmp[j],"\n",1);
          write(fd_result,"\n",1);
        }

        close(p[0]);
        close(fd_tmp[j]);
        j++;
        write(fd_result,"<<<\n",4);
      }else{
        perror("NAO ABRIU O FICHEIRO");
        _exit(1);
      }
    }else{
      remove("result");
      perror("COMANDO IMPOSSIVEL DE EXECUTAR");
      _exit(1);
    }
```

Aqui apresentamos o *if* que faltava no ciclo *while* apresentado anteriormente, ou seja, entramos nesta condição quando possuímos uma linha com *$* seguido *numero* seguido de .

Capítulo 4

Testes

Com os seguintes testes já é possível observar todas as funcionalidades que implementámos a funcionar. Ou seja, é possível observar que a execução de programas, o re-processamento do notebook, a deteção de erros/interrupção da execução e o acesso a resultados de comandos anteriores estão a funcionar.

- **Teste 1:**

```
Este comando lista os ficheiros:
$ ls
Agora podemos ordenar estes ficheiros:
$| sort
E escolher o primeiro:
$2| head -5
Tail do anterior:
$| tail -1
Grep do ls:
$4| grep notebook
```

- **Resultados Teste 1:**

```
$ ls
>>>
0.txt
1.txt
2.txt
3.txt
4.txt
5.txt
enunciado-so-2017-18.pdf
notebook
notebook.c
README.md
result.txt
teste
teste1.txt
teste2.txt
teste3.txt
teste4.txt
teste.txt
```

```
<<<
$| sort
>>>
0.txt
1.txt
2.txt
3.txt
4.txt
5.txt
enunciado-so-2017-18.pdf
eter15  4970 May  8 20:35 notebook.c
notebook
notebook.c
README.md
result.txt
-rw-r--r-- 1 jeter15 jeter15    10 May  8 20:35 README.md
-rw-r--r-- 1 jeter15 jeter15   122 May  8 20:35 teste.txt
-rwxr-xr-x 1 jeter15 jeter15    12 Jun  2 16:49 result.txt
-rwxr-xr-x 1 jeter15 jeter15    62 May  8 20:35 teste
teste
teste1.txt
teste2.txt
teste3.txt
teste4.txt
teste.txt
<<<
$2| head -5
>>>
0.txt
1.txt
2.txt
3.txt
4.txt
<<<
$| tail -1
>>>
total 116
<<<
$4| grep notebook
>>>
notebook
notebook.c
eter15  4970 May  8 20:35 notebook.c
<<<
```

- **Teste 2:**

```
Este comando lista os ficheiros:
$ ls -l
Vamos buscar as 3 primeiras linhas do anterior:
$| head -3
Agora podemos ordenar estes ficheiros:
$| sort
Vamos buscar as 2 primeiras linhas do anterior:
$| head -2
Vamos buscar a primeira linha de 2 comandos anteriores:
$2| head -1
Agora podemos ordenar estes ficheiros:
$5| sort
```

- **Resultados Teste 2:**

```
$ ls
>>>
0.txt
1.txt
2.txt
3.txt
4.txt
5.txt
enunciado-so-2017-18.pdf
notebook
notebook.c
README.md
result.txt
teste
teste1.txt
teste2.txt
teste3.txt
teste4.txt
teste.txt
<<<
$| sort
>>>
0.txt
1.txt
2.txt
3.txt
4.txt
5.txt
enunciado-so-2017-18.pdf
eter15  4970 May  8 20:35 notebook.c
notebook
notebook.c
README.md
result.txt
-rw-r--r-- 1 jeter15 jeter15    10 May  8 20:35 README.md
-rw-r--r-- 1 jeter15 jeter15   122 May  8 20:35 teste.txt
-rwxr-xr-x 1 jeter15 jeter15    12 Jun  2 16:49 result.txt
-rwxr-xr-x 1 jeter15 jeter15    62 May  8 20:35 teste
```

```
teste
teste1.txt
teste2.txt
teste3.txt
teste4.txt
teste.txt
<<<
$2| head -5
>>>
0.txt
1.txt
2.txt
3.txt
4.txt
<<<
$| tail -1
>>>
total 116
<<<
$4| grep notebook
>>>
notebook
notebook.c
eter15  4970 May  8 20:35 notebook.c
<<<
```

Capítulo 5

Conclusão

Este trabalho foi bastante importante e enriquecedor uma vez que permitiu que todos os elementos do grupo compreendessem e assimilassem melhor os conceitos lecionados na unidade curricular durante o semestre.

Foi então possível compreender melhor os mecanismos para a criação de processos bem como o seu modo de funcionamento e de comunicação com outros, neste caso através de *pipes* como foi demonstrado anteriormente na elaboração do controlador. Foi também útil para aprendemos mais sobre o sistema operativo utilizado (Linux), percebendo como este poderia ser explorado.