FSO

Trabalho prático 1

Shell

No trabalho prático 1(aka) Shell, utilizámos as seguintes bibliotecas da linguagem C:
stdio.h, stdlib.h, string.h, unistd.h, sys/wait.h.

Com 2 array's para de apontadores do tipo char argv e argv2 com o mesmo tamanho, um array de char's chamado linha e a variável status que guarda o valor retornado pelo processo filho.

Temos métodos do estilo booleano(não sao, são inteiros mas funcionam como booleanos na nossa linha de pensamento) muito práticos para saber quando dado acontecimento ou dado está presente.

*Funções int check(char *myargv[]) e int checkPipe(char *myargv[])*
Um deles é o método check que recebe, como argumento um array de apontadores do tipo char e percorre esse vector à procura do & se o encontrar coloca NULL no endereço "i" do vector e retorna 0 senão retorna 1.Outro método pseudo booleano é o checkPipe que recebe como argumento um array de apontadores do tipo char, se ele encontra "|" coloca null como valor na posição i desse vector, porque haverá pipe, e ao mesmo tempo vai contruir um segundo argv, chamado myargv2, com os valores restantes do original, colocando null no final porque o ciclo acaba sem o fazer após tudo isso sendo um método do tipo int tem que retornar um inteiro logo se há pipe retorna 1 senão retorna 0.

Função void runcmd(char in[])
Para tratamento da suposta "string" temos um método chamado runcmd que recebe como argumento um array de char's, vai "partir" cada caracter, com delimitadores(" " e "\n") e quando no final o token seguinte apontar para null coloca-se o valor da mesma posiçao no vector a null.

*Função void runcommand(char *myargv[])*
Temos um método usado para quando o comando é externo chamado runcommand que é um void e tem como argumento um array de apontadores do tipo char, para além de executar os comandos externos vai também executar processos em background e foreground, o processo cria um outro processo filho, se a função fork retornar 0, senao aborta, a variavel temp vai ter 0 ou 1 conforme tenha encontrado ou não & no vector, se não encontrou então vai correr o comando em background.

Função void runWithPipe()
Para execução de comandos temos ainda o runWithPipe para quando utilizamos um pipe com dois comandos. Ao contrário do método runcommand este vai utilizar canais para que o 2º comando vá ler os valores do 1º.Utilizamos a chamada ao sistema pipe e a fork com as respectivas condições e "abortions"caso tenha ocorrido algum erro, e quando se consegue fazer o processo filho e o neto estes vão chamar métodos que têm que estar implimentados acima deste código para não haver erro de compilação, esses 2 métodos trataram de redireccionar os canais.

Funções void filho1(int p[]) e void filho2(int p[])
Método filho1 é um método void e é 1º metodo que é chamado, supostamento o filho, vai criar um novo descritor com o valor do canal de escrita, e fecha o velho canal, fecha também o canal inútil de leitura, e o de escrita porque já fizems o que queriamos.
Método filho2 é também um método do tipo void, e o 2º a ser chamado chamado, supostamento o neto, vai criar um novo descritor com o valor do canal de leitura, com os mesmos valores que o original fecha o canal inutil, e o de leitura porque já fizems o que queriamos.

Função int main ()
Na main, guarda-se espaço em disco do tamanho do char, chama-se a funçao getcwd() que guarda na variável temp o path até à directoria actual.Depois opera-se o vector com a função runcmd() , a seguir acha-se se há um pipe se houver executa-se o método runWithPipe, se não houver opera-se aqui, senão for reconhecido aqui é porque é um comando externo.No fim liberta-se o estado ocupado.

Trabalho realizado por:
Pedro Miranda, nº 28289
Tiago Silva Ferreira, nº28165