`simgrep`: versão simplificada de "`grep`"

Metas de aprendizagem

Completando com sucesso o trabalho, os alunos demonstram conhecer e saber utilizar a interface programática de UNIX para:

- criar novos processos;
- fazê-los intercomunicar por sinais;
- percorrer um sistema de ficheiros e dele obter informações.

Descrição geral

A parte de programação do trabalho consiste na escrita de uma versão (muito) simplificada do utilitário de Unix "`grep`" [1] que permite encontrar padrões de texto em ficheiros[2], podendo também percorrer diretórios.

O programa aqui desenvolvido, `simgrep`, deve ser capaz de reproduzir os resultados apresentados pela invocação de `grep` para os mesmos argumentos. Além disso, deve exibir duas funcionalidades que não existem no `grep` e que são descritas mais à frente (possibilidade de interrupção de execução e geração de registos de execução).

A estrutura do programa é deixada ao critério do projetista, mas exige-se o cumprimento de certos requisitos, também apresentados mais à frente.

Exemplos de invocação com `grep`:

```
shell$ grep chair 1.txt
down in his armchair and closing his eyes.
The man sprang from his chair and paced up and down the room in
chair.

shell$ grep -i -w Chair 1.txt
The man sprang from his chair and paced up and down the room in
chair.

shell$ grep -n -i -w Chair 1.txt
303:The man sprang from his chair and paced up and down the room in
521:chair.

shell$ grep -c chair 1.txt
3

shell$ grep -l -r chair ../AdvSherlockHolmes
../AdvSherlockHolmes/1.txt
../AdvSherlockHolmes/7.txt
......
```

1 `man grep`: grep (...) - print lines matching a pattern

2 como primeira simplificação, considerar que `simgrep` será apenas usado para analisar ficheiros de texto.

Requisitos funcionais

O programa deve comportar-se como o comando `grep` tipicamente encontrado nos sistemas Unix, mas apenas num subconjunto das suas funcionalidades. A sua linha de comando genérica será :

```
shell$ simgrep [options] pattern [file/dir]
```

Deve ser capaz de reconhecer as seguintes opções de linha de comando e agir consequentemente:

- `-i` ignorar o tipo das letras (maiúsculas/minúsculas) do padrão a procurar;
- `-l` mostrar apenas o nome dos ficheiros onde estiver o padrão a procurar;
- `-n` indicar também o número das linhas onde estiver o padrão a procurar;
- `-c` indicar em quantas linhas o padrão a procurar é encontrado;
- `-w` o padrão a procurar deve formar uma palavra completa. (Ver como `grep` define uma palavra!)
- `-r` procurar o padrão em todos os ficheiros abaixo da árvore do diretório indicado.

Notar também que **`simgrep`**, à semelhança de **`grep`**, deve ler a entrada padrão (**`stdin`**) no caso de na linha de comando estar omisso o ficheiro onde o padrão deve ser procurado.

Funcionalidades adicionais

Interrupção pelo utilizador: estando **`simgrep`** em execução, se se carregar em `CTRL+C`, todo o programa deve interromper a operação e colocar ao utilizador a pergunta: «`Are you sure you want to terminate the program? (Y/N)`». Se a resposta for `Y` (ou `y`), todo o programa termina sem mais delongas; se for `N` (ou `n`), todo o programa prossegue como se nada tivesse acontecido.

Geração de registos de execução: na sequência da invocação de **`simgrep`**, o primeiro processo deve criar um ficheiro (se ele já não existir!) que irá servir para todos os processos participantes na operação registarem os principais eventos que desencadearem ou de que receberam notificação (abertura e fecho de ficheiros, receção e emissão de sinais, etc.). O nome desse ficheiro é passado aos processos (inclusive ao inicial) por via de uma variável de ambiente, **`LOGFILENAME`**, a criar pelo utilizador; qualquer dos processos participantes usa o ficheiro, acrescentando-lhe informação, linha a linha[3]. O formato de linha do ficheiro de registos será:

- *`inst - pid - act`*
  em que
  - *`inst`* é o instante de tempo imediatamente anterior ao registo, medido em milisegundos e com 2 casas decimais, e tendo como referência o instante em que o programa começou a executar;
  - *`pid`* é o identificador do processo, com espaço fixo para 8 algarismos;
  - *`act`* é a descrição do evento: "`COMANDO grep -c chair 1.txt`", "`SINAL USR1`", "`SINAL USR2 para 12345678`", "`ABERTO 1.txt`", "`FECHADO [fich.html]`", etc.
    Em itálico indica-se a informação que irá variar; por outro lado, outras descrições de eventos julgados relevantes podem ser acrescentadas à lista mostrada.

A informação resultante no ficheiro de registos será útil para a verificação do bom funcionamento do programa, após a sua conclusão.

Requisitos arquiteturais

Apesar de, como foi dito, a estrutura do programa ser deixada a cargo de quem o vai escrever, há alguns requisitos arquiteturais que são exigidos. O programa deve:

- criar um processo por cada diretório a analisar;
- no caso da opção '**`-r`**', comportar-se de forma recursiva: o que o primeiro processo fizer no diretório inicial, será repetido pelos processos descendentes, mas sobre os diretórios que lhes forem atribuídos;
- no caso da opção '**`-r`**', percorrer cada diretório distinguindo os tipos de ficheiro nele contidos:

3 ao final do ficheiro – modo *append*!

- a um ficheiro normal, será aplicada a pesquisa do padrão de acordo com as opções estipuladas na linha de comando;
- a um (sub-)diretório, corresponderá a criação de um processo descendente idêntico, que repetirá o aqui descrito a esse (sub-)diretório.

Produto final

O trabalho total consiste na produção de um ficheiro compacto, que inclui o código-fonte com o programa desenvolvido e um *makefile* preparado para facilitar a geração do executável[4]. O compacto é identificado com um nome do tipo `TxGyy.tar.gz`, onde **`x`** e **`yy`** são o número da turma e do grupo, respetivamente.

Avaliação

Será efetuada através de testes simples de execução, em que os resultados produzidos pelo programa `simgrep` desenvolvido serão comparados com os resultados produzidos por `grep` em condições idênticas. (Notar que, eventualmente, poderá haver a necessidade de ordenar as linhas impressas.)

O funcionamento do requisito relativo à reação do programa relativamente à sua interrupção inesperada através da ativação de `CTRL+C`, também será experimentado.

4 Não esquecer importantes opções de compilação, tal como `Wall`, `Wextra` e, até, `Werror`.