TRABALHO PRÁTICO Nº 1

Ferramenta de Análise Forense Digital

Metas de aprendizagem

Completando com sucesso o trabalho, os alunos demonstram conhecer e saber utilizar a interface programática de Unix/Linux 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 um utilitário de análise forense a diretórios e ficheiros em ambiente Unix/Linux.

Tal utilitário deve ser capaz de percorrer um sistema de ficheiros, recolhendo ou criando informação sobre os ficheiros encontrados, e apresentando-a num formato especificado. O utilizador escolhe o diretório de partida (ou, até, um simples ficheiro), o local onde a informação colecionada é colocada (saída padrão ou ficheiro) e a geração de registos de execução (*logs*). Adicionalmente, o utilitário deve lidar com a possibilidade de interrupção de execução, a comando do utilizador.

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

Informação Adicional

O sistema GNU/Linux fornece um conjunto de comandos que auxiliam a obtenção de informação sobre ficheiros e a geração de novas informações sobre eles. Por exemplo, o comando **`file`** fornece informação sobre o tipo de ficheiro. Comandos como **`md5sum`**, **`sha1sum`** e **`sha256sum`** calculam sumários criptográficos[1] dos ficheiros indicados como parâmetro. Alguns comandos são exemplificados de seguida (note-se o formato hexadecimal resultante das operações criptográficas):

```
shell$ file hello.c
hello.c: C source, ASCII text

shell$ md5sum hello.txt
4ff75ff116aad93549013ef70f41e59c  hello.txt

shell$ sha256sum hello.txt
7605bfc397c3d78b15a8719ddbfa28e751b81c6127c61a9c171e3db60dd9d046  hello.txt
```

1 Um sumário criptográfico, em inglês *cryptographic hash*, permite identificar um ficheiro de qualquer tipo ou tamanho, através de uma sequência de bits, também designada "impressão digital" do ficheiro. Como se vê pelos exemplos, diferentes algoritmos geram impressões digitais diferentes.
Esta identificação de ficheiros, guardada numa dada altura, facilita a verificação de alterações (eventualmente maliciosas) dos ficheiros em alturas posteriores.

Exemplos de Invocação

O utilitário a desenvolver, denominado **`forensic`**, pode ser invocado de diversas maneiras, como os exemplos seguintes ilustram.

```
shell$ forensic hello.txt
hello.txt,ASCII text,100,rw,2018-01-04T16:30:19,2018-01-04T16:34:13
```

```
shell$ forensic -h sha1,sha256 hello.txt
hello.txt,ASCII text,100,rw,2018-01-04T16:30:19,2018-01-
04T16:34:13,d2d29b8b66e3ef44f3412224de6624edd09cdb0c,7605bfc397c3d78b15a8719ddbf
a28e751b81c6127c61a9c171e3db60dd9d046
```

```
shell$ forensic -r folder
file1.txt,ASCII text,950,rw, 2018-01-04T15:30:19,2018-01-04T15:34:38
file2,ELF 64-bit LSB shared object,100,rw,2017-01-04T16:30:19,2017-01-
04T16:34:29
subfolder/file2,ELF 64-bit LSB shared object,192,rw, 2017-11-04T16:30:19,2017-
11-04T16:34:09
```

```
shell$ forensic -h md5 -o output.txt -v hello.txt
Data saved on file output.txt
Execution records saved on file ...
```

Requisitos Funcionais

O utilitário, como exemplificado, deve colecionar mais ou menos informação sobre um ficheiro ou sobre os ficheiros contidos numa diretoria, de acordo com a vontade do utilizador. A sua linha de comando genérica será:

`forensic [-r] [-h [md5[,sha1[,sha256]]] [-o <`*`outfile`*`>] [-v] <file|dir>`

As opções de linha de comando possíveis são a seguir explicadas.

- **-r** : analisar todos os ficheiros do diretório indicado e de todos os seus subdiretórios.
- **-h** : calcular uma ou mais "impressões digitais" dos ficheiros analisados. Pode pedir-se os algoritmos MD5, SHA1 ou SHA256; querendo-se mais do que um, separar os identificadores por vírgulas.
- **-v** : gerar ficheiro com os registos de execução, conforme explicado mais à frente. O nome do ficheiro é obtido da variável de ambiente `LOGFILENAME`.
- **-o** : armazenar os dados da análise no ficheiro indicado (e não na saída padrão). O ficheiro terá, naturalmente, o formato CSV (*comma-separated values*), devido à forma como a informação é apresentada.

Apresentação dos resultados

Os resultados da análise efectuada devem ser apresentados na saída padrão ou num ficheiro designado, no seguinte formato de linha, cada uma correspondente a um ficheiro analisado:

```
file_name,file_type,file_size,file_access,file_created_date,file_modification_da
te,md5,sha1,sha256
```

De notar que pode acontecer nem todos os campos serem utilizados, pois os relativos às somas criptográficas dependem da escolha do utilizador. As somas criptográficas são em hexadecimal.

As datas (que incluem os tempos) devem ser apresentadas num formato ISO 8601, `<date>T<time>`. O seguinte exemplo ilustra o conteúdo de uma linha.

```
hello.txt,ASCII text,100,rw,2018-01-04T16:30:19,2018-01-04T16:34:13,
4ff75ff116aad93549013ef70f41e59c,d2d29b8b66e3ef44f3412224de6624edd09cdb0c,7605bf
c397c3d78b15a8719ddbfa28e751b81c6127c61a9c171e3db60dd9d046
```

Geração de registos de execução

Na sequência da invocação do utilitário com a opção **-v**, deve ser criado um ficheiro (se ele já não existir) onde serão registados os principais eventos relativos às operações de coleta de informação (novo ficheiro encontrado, receção e emissão de sinais, etc.).

O nome desse ficheiro é passado ao programa por via de uma variável de ambiente, `LOGFILENAME`, a criar pelo utilizador. Qualquer dos processos participantes na operação do programa acede ao ficheiro, acrescentando-lhe informação, linha a linha[2].

O formato de linha do ficheiro de registos será:

inst – pid – act

- **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 que faz o registo da linha, com espaço fixo para 8 algarismos;
- **act**: é a descrição do evento. Alguns exemplos: "`COMMAND` *`forensic -r ./folder`*", "`SIGNAL` *`USR1`*", "ANA`LIZED` *`1.txt`*", etc. Em itálico indica-se a informação que irá variar; outras descrições de eventos julgados relevantes podem ser acrescentadas.

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

Funcionalidades Adicionais

Interrupção pelo utilizador

Estando o utilitário em execução, se se carregar em **CTRL+C**, todo o programa deve interromper a sua operação, terminando quaisquer processos associados, mas tendo o cuidado de se terminar todas as operações pendentes (por exemplo, completar uma linha de registo em curso ou eliminar eventuais ficheiros temporários).

Informação de progresso

No caso de a opção de escrita para um ficheiro (**-o**) estar ativada (não aparecendo informação dos resultados do programa na saída padrão), deve ser emitido um sinal **SIGUSR1** sempre que um novo diretório é encontrado e o seu processamento é iniciado e, de forma similar, um sinal **SIGUSR2** para um novo ficheiro.

Uma rotina de tratamento daqueles sinais irá contabilizando os sinais recebidos, por forma a se saber quantos ficheiros e quantos directórios são encontrados. No caso de notificação de um novo directório, a rotina colocará na saída padrão uma mensagem do tipo:

```
New directory: n/m directories/files at this time.
```

em que ***n*** e ***m*** são, obvia e respectivamente, o número de directórios e de ficheiros contabilizados até ao momento.

2 ao final do ficheiro – modo *append*!

Interessante será verificar se o número final apresentado corresponde efectivamente ao número total de ficheiros e directórios analisados!

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, que irá colecionar informações sobre os ficheiros do diretório; esse processo executará concorrentemente com eventuais outros processos participantes;
- 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.

Plano de Trabalho

De forma a tornar o desenvolvimento do utilitário mais modular, deve ser seguido o seguinte plano de trabalhos:

1. Receber, tratar e guardar os argumentos e variáveis de ambiente.
2. Extrair a informação solicitada de apenas um ficheiro e imprimi-la na saída padrão de acordo com os argumentos passados.
    - Efetuar o mesmo procedimento mas agora, implementando a operação da opção '**`-o`**' (escrita no ficheiro designado).
3. Repetir o passo anterior para todos os ficheiros de um diretório.
4. Criar um processo filho quando for encontrado um diretório, capaz de vir a repetir as funções do pai, efetuando um trabalho semelhante ao do passo anterior.
5. Pôr a funcionar a análise recursiva de uma árvore de diretórios.
6. Adicionar as funcionalidades de registo (*log).*
7. Adicionar a funcionalidade de tratamento do sinal associado ao **`CTRL+C`**.
8. Adicionar a funcionalidade de emissão e tratamento dos sinais **`SIGUSR1`** e **`SIGUSR2`**.

O trabalho deverá ser realizado em grupo, sendo cada grupo constituído por 3 estudantes.

Sempre que os grupos de alunos concluírem um ou mais pontos deste plano podem fazer uma demonstração durante a aula prática, de forma a receber aconselhamento/comentários sobre o estado do projeto até ao momento.

Produto final

Deve ser produzido um ficheiro compacto, que inclua o código-fonte do programa desenvolvido e um *makefile* preparado para facilitar a geração do executável[3]. 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, e, ao ser descompactado, cria uma pasta designada **`TxGyy`** com os ficheiros relativos ao trabalho.

Avaliação

NOTA: Apesar deste 1º trabalho prático não ser classificado, não sendo tido em conta para a classificação de frequência, o seu desenvolvimento, entrega e apresentação (quando solicitada) são condições necessárias para a obtenção de frequência.

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