

Projecto Integrado de Laboratórios de Informática I

Universidade do Minho

8201N6 — Lab. de Informática I
LEI — Licenciatura de Engenharia Informática
Ano lectivo de 2011/12

Dezembro de 2011

Conteúdo

1	Objectivos	1
2	Catálogo multimédia	2
3	Material	2
4	Tecnologias e ferramentas	2
5	O que há a fazer	3
A	Exemplo ilustrativo	6
B	Reconhecimento de texto em Haskell	10
C	Instalação das ferramentas	11
D	Kit Multimédia	12

1 Objectivos

Este projecto tem como objectivo uma consolidação e visão integrada do uso da shell do Unix, do sistema \LaTeX para preparação de texto e da linguagem Haskell para programação funcional. Do projecto consta o desenvolvimento de um sistema de software de pequena escala.

O projecto parte de uma base (“kit”) já desenvolvida pelo regente e monitores da disciplina, obrigando os alunos a estudar código escrito e ferramentas desenvolvidas por terceiros. O objectivo é simular o desenvolvimento de um projecto real de software, onde é frequentemente necessário lidar com a manutenção e integração de código já desenvolvido.

Do tema escolhido (tratamento de conteúdos multimédia) resulta também do trabalho uma acção pedagógica na sensibilização dos alunos para os regimes e mecanismos de *copyright*, que é um tema da maior actualidade.

2 Catálogo multimédia

O projecto consiste na catalogação de todos os ficheiros multimédia presentes numa dada directoria, sob a forma de uma pequena base de dados na qual (recorrendo à linguagem de programação Haskell) se possam efectuar operações como:

- visualizar o registo de um dado ficheiro multimédia;
- fazer seleções e ordenações de conteúdos relevantes (eg. seleccionar músicas de um dado artista);
- inspeccionar regimes de *copyright*;
- construir catálogos em PDF da informação seleccionada usando Haskell para exportar essa informação para L^AT_EX, que se encarregará de fazer o respectivo “pretty-printing”.

Como valorização, o sistema poderá ainda permitir:

- adicionar *frames* ou imagens aos conteúdos da alínea anterior;
- integrar conteúdos multimédia directamente nos ficheiros PDF, incluindo vídeo, música, *screen capture* de vídeos e outros ficheiros presentes no *kit multimedia* que é fornecido.

3 Material

O presente documento (enunciado) é o principal conteúdo para a realização deste trabalho e como tal deve ser cuidadosamente estudado. O material auxiliar obtém-se descompactando o ficheiro

li11112mp.zip

do qual emergem, para além do corrente enunciado, as bibliotecas Haskell

- Dir.lhs
- TeX.lhs

que vão ser necessárias para o projecto.

O kit multimedia que deverá ser usado para testes e exemplos encontra-se descrito e acessível no apêndice D.

4 Tecnologias e ferramentas

Shell. Para além de os alunos terem de estar familiarizados com os comandos básicos da shell, estudados nas aulas, terão de aprender a usar duas ferramentas novas para extracção de meta-dados de formatos multimédia e para a recodificação e extracção de conteúdos. Essas ferramentas serão, respectivamente, `ExifTool` e `FFmpeg`¹.

Sendo um dos objectivos do trabalho familiarizar os alunos com as técnicas de instalação de pacotes de software tal como são distribuídos em contexto industrial, devem seguir-se as instruções de instalação e exemplos de utilização que constam do apêndice C.

¹Opcionalmente poderá também ser necessária a utilização da ferramenta ImageMagick.

Haskell. Será nesta linguagem que será desenvolvida a maior parte do projecto. Chamando as ferramentas acima referidas ao nível da shell, o programa a desenvolver em Haskell deverá começar por recolher informação multimédia sobre que ficheiros a processar (chamando `exiftool` para esse efeito) e guardar essa informação num formato (tipo de dados) apropriado, a desenvolver.

Todas as funcionalidades pretendidas serão então construídas sobre esse tipo de dados, incluindo a geração de \LaTeX para produção dos catálogos em PDF. Para este efeito foi desenvolvido pela equipa docente uma representação em Haskell de um subconjunto do \LaTeX — designado “micro \LaTeX ” ($\mu\text{\LaTeX}$) — que está disponível na biblioteca `TeX.lhs`.

Esta e outras bibliotecas deverão ser cuidadosamente estudadas antes da realização do trabalho. No apêndice A ilustra-se o método a seguir com um exemplo bastante mais simples — mostrar o conteúdo de directorias via comando `ls`. Recomenda-se aos alunos que sigam este exemplo e o adaptem ao que é pretendido.

\LaTeX . Este sistema de preparação de conteúdos desempenha um papel central no projecto, pois é recorrendo a ele que o PDF dos catálogos a produzir será gerado. O \LaTeX assim produzido deverá incluir os conteúdos multimédia que serão construídos através do Haskell e das ferramentas auxiliares de linha de comandos. Lidar com boa formatação para a exibição correcta dos conteúdos será essencial à vertente multimédia do projecto.

Para embeber multimédia em PDFs gerados pelo \LaTeX pode ser usado o *package* `movie15` e o comando `\includemovie` para vídeos e sons². O package pode ser instalado usando o manager do TeX Live ou do MikTeX, dependendo de que se tiver instalado. No caso do MacTeX, o package já está incluído na distribuição.

Veja-se, a título de exemplo, o seguinte fragmento de \LaTeX ,

```
\begin{figure}[htb]
\begin{center}
{\Huge Trouble}\\
Music: Kevin MacCleod.
\includemovie[open,externalviewer]{}{}{Trouble.mp3}
\end{center}
\end{figure}
```

e o efeito que produz:

Trouble
Music: Kevin MacCleod.

Se carregarem no ícon verão que o *player* que tiverem pré-definido para o formato `mp3` será activado, fazendo soar um pequeno fragmento de música de 17 segundos.

5 O que há a fazer

Primeira parte:

1. Ler muito bem o apêndice A, construindo e testando o exemplo dado até ao fim.

² É importante saberem que a visualização desses PDFs enriquecidos requer o Adobe Acrobat Reader.

2. Instalar as ferramentas a usar ao nível da *shell* de acordo com as instruções do apêndice C e praticar um pouco com elas, ao nível da linha de comandos num terminal.
3. Estudar as bibliotecas auxiliares `TeX` e `Dir`. Estes ficheiros estão escritos em *literate Haskell* — uma mistura de \LaTeX e Haskell (daí a extensão `lhs`) que permite escrever num só ficheiro um programa e a sua documentação³.
4. Adaptar o exemplo do apêndice A à invocação da ferramenta `exiftool`, chamando-a para cada ficheiro obtido a partir de `ls`. Para isso é preciso construir uma nova biblioteca — chamemos-lhe `Exif` — que ofereça tipos em Haskell para registar a informação que `exiftool` extrai dos ficheiros multimedia (eg. som, vídeo).

Importante: para simplificar esta tarefa deve ser cuidadosamente estudado o apêndice B.

5. Essa biblioteca deverá oferecer também uma função de “pretty-printing” desses tipos para \LaTeX , tal como é feito em `Dir`.

Segunda parte:

1. Uma vez gerado o programa `dat.hs` na primeira parte, programar as funções de ordenação e selecção de conteúdos, como se sugere no apêndice A.
2. Instanciar os tipos para som e vídeo, programados na biblioteca `ExifTool` na classe `Show` por forma a permitir mostrar, ao terminal, cada registo da forma seguinte:

```
+-----+-----+
| Filename | I Knew a Guy.mp3 |
+-----+-----+
| Size     | 5.8 MB             |
+-----+-----+
| Ftype    | MP3                 |
+-----+-----+
| Copyright | copyright           |
+-----+-----+
| Title     | I Knew a Guy        |
+-----+-----+
| Artist    | Kevin MacLeod       |
+-----+-----+
| Year      | 2006                 |
+-----+-----+
```

se áudio, e da forma

```
+-----+-----+
| Filename | 4th_dan-trailer.mp4 |
+-----+-----+
| Size     | 22 MB                |
+-----+-----+
| Ftype    | MP4                  |
+-----+-----+
| Copyright | copyright             |
```

³ Mais informação em <http://people.cs.uu.nl/andres/lhs2tex>.

Width	1280
Height	720

se vídeo.

3. Complementar a biblioteca TeX com mais construções do L^AT_EX, a saber: declaração de arrays, minipáginas; comandos de integridade referencial: label e ref.
4. Exportar o catálogo para PDF registrando sempre a informação de copyright⁴.

Do código desenvolvido devem apresentar-se versões interpretadas e compiladas recorrendo, se necessário, a makefiles.

Valorização. Havendo tempo e desejando-se aprender mais, escolher entre as seguintes sugestões:

1. Catálogos especiais
 - Extrair capas de álbuns dos MP3 usando o `exiftool`.
 - Produzir catálogos PDF com informação multimédia adicional (vídeos, sons, imagens), adicionando (quando necessário) novas primitivas ao μ L^AT_EX (módulo `Exp`).
 - Gerar catálogos como *booklets* para distribuição electrónica (PDF e HTML) e impressão.
 - Gerar catálogos para impressão, excluindo informação multimédia irrelevante (sons e vídeos).
 - Personalizar informação nos catálogos.
2. FFmpeg
 - Usar o FFmpeg para extrair frames de vídeos e colocá-los no PDF gerado.
 - Criar uma *amostra* (“sample”) de 10s de um vídeo do kit multimédia.
 - Colocar legendas na amostra, usando o formato SRT e o formato de ficheiro MKV.
 - Colocar no PDF a amostra associada aos frames retirados do filme.
3. ImageMagick
 - Adicionar texto relevante a imagens (autor, copyright).
 - Normalizar o tamanho das imagens e criar um slideshow com o FFmpeg.
 - Adicionar som ao slideshow.

Relatório. Deste trabalho deverá ser feito um pequeno relatório que deverá ser entregue ao monitor do turno de cada grupo, bem como o código desenvolvido, de acordo com instruções a publicar na página da disciplina.

⁴Atenção que alguns ficheiros podem não ter a informação de *copyright* nos seus meta-dados.



Music: Kevin MacLeod.

Figura 1: Exemplo das valorizações.

Anexos

A Exemplo ilustrativo

Suponhamos que o que pretendemos é fazer “pretty-printing” de uma directoria, tão simples como isso. Se em Haskell se escrever

```
import System.Cmd
```

importa-se uma biblioteca que permite aceder à shell para esse efeito:

```
system "ls -l > txt.txt"
```

guardando-se neste caso o resultado no ficheiro `txt.txt`. A função `mymain` seguinte mostra como ler esse ficheiro e usar funções elementares de processamento de “strings” para obter a mesma informação numa lista de listas:

```
mymain =  
  do {  
    system "ls -l > txt.txt" ;  
    a <- readFile "txt.txt" ;  
    let text = map words (lines a)  
    in print text  
  }
```

Por exemplo, se a directoria contiver os ficheiros

```
-rw-r--r--  1 jno  jno  171 Dec  1 12:27 j.hs  
-rw-r--r--  1 jno  jno  257 Dec  1 12:27 txt.txt
```

o resultado de correr `mymain` será:

```
[["total","8"],  
 ["-rw-r--r--","1","jno","jno","171","Dec","1","12:27","j.hs"],  
 ["-rw-r--r--","1","jno","jno","257","Dec","1","12:29","txt.txt"]]
```

Repare-se agora na biblioteca `Dir` fornecida, onde os tipos `Dir` e `File` representam directorias e ficheiros em Haskell, respectivamente. Estendendo a nossa função acima da forma seguinte,

```

mymain =
  do {
    system "ls -l > txt.txt" ;
    a <- readFile "txt.txt" ;
    let text = map words (lines a)
        total = head text
        files = map seq9toFile (tail text)
        dir = Dir (read (total !! 1)) files
    in print dir
  }

```

(onde

```

seq9toFile s =
  File (s!!0) (s!!1) (s!!2) (s!!3) (read(s!!4)) (s!!5) (s!!6) (s!!7) (s!!8)

```

converte listas de pelo menos nove elementos num objecto de tipo File) mymain dará agora como resultado a mesma informação, mas agora no formato Dir:

```

Dir {total = 32, files = [File {perms = "-rw-r--r--", nrlnk = "1",
owner = "jno", group = "jno", size = 2014, month = "Dec", day =
"1", time = "13:19", name = "Dir.lhs"}, File {perms = "-rw-r--r--",
nrlnk = "1", owner = "jno", group = "jno", size = 4796, month =
"Dec", day = "1", time = "13:19", name = "TeX.lhs"}, File {perms
= "-rw-r--r--", nrlnk = "1", owner = "jno", group = "jno", size
= 574, month = "Dec", day = "1", time = "13:29", name = "j.hs"},
File {perms = "-rw-r--r--", nrlnk = "1", owner = "jno", group =
"jno", size = 0, month = "Dec", day = "1", time = "13:32", name
= "txt.txt"}]}

```

(Note-se a adição das duas bibliotecas TeX e Dir, que agora são necessárias.)

O passo seguinte consiste em fazer “pretty printing” desta informação para L^AT_EX. Repare-se que a biblioteca Dir já contém funções para isso, em particular dirDir2TeX e prettyprint. Assim, se em lugar de

```
in print dir
```

escrevermos

```
in print (dirDir2TeX dir)
```

obteremos a mesma informação, agora já em μ L^AT_EX:

```

Main> mymain
TeXs [Cmd (TeXCmd {op = "begin", str = [STR "quote"]}),TeXs [TeXs
[STR "Total: 32",TeXs [Cmd (TeXCmd {op = "begin", str = [STR
"description"]}),TeXs [TeXs [TeXs [Cmd (TeXCmd {op = "item", str
= [Opt {t = STR "File: "}]})],TeXs [STR "Dir.lhs",TeXs [Cmd (TeXCmd
{op = "begin", str = [STR "quote"]}),TeXs [TeXs [Cmd (TeXCmd {op =
.
.
.
.
= [STR "description"]})]]],Cmd (TeXCmd {op = "end", str = [STR
"quote"]})]

```

(Omita-se texto para poupar espaço.) Mas nós queremos mais: queremos essa informação em sintaxe concreta do L^AT_EX, para a podermos processar e ver em PDF.

Para isso veja-se o efeito de usar prettyprint em lugar de dirDir2TeX, reescrevendo-se a mesma linha da forma seguinte:

```
in print (prettyprint dir)
```

Obter-se-á (de novo omitindo texto para poupar espaço):

```
"\n\documentclass{article}\n\begin{document}\n\begin{quote}Total:
32\n\begin{description}\n\item[File: ]Dir.lhs\n\begin{quote}\n
\begin{tabular}{|l|l|}\n\hline Attribute&Data\n\hline
perms&-rw-r--r--\n\hline nrlnk&1\n\hline
.
.
.
\n\end{tabular}\n\end{quote}\n\end{description}\n\end{quote}\n
\end{document}"
```

Falta-nos apenas o passo final: gerar um ficheiro com este conteúdo e processá-lo em \LaTeX . Podemos fazer isso no próprio Haskell que estamos a escrever, o que nos conduz à versão final da função `mymain`:

```
mymain =
  do { system "ls -l > txt.txt" ;
      a <- readFile "txt.txt" ;
      let text = map words (lines a)
          total = head text
          files = map seq9toFile (tail text)
          dir = Dir (read (total !! 1)) files
          hs = "import Dir\nimport TeX\nimport System.Cmd"
              ++ "\ntoTeX= do { writeFile \"out.tex\" $ prettyprint dat ;"
              ++ "system \"pdflatex out\"}"
              ++ "\ndat="
              ++ show dir
      in do { writeFile "dat.hs" hs ;
              putStrLn "\nOk.\nNow load \'dat.hs\' and run \'toTeX\'"
            }
  }
```

Repare-se que a função termina escrevendo um programa em Haskell⁵ no ficheiro `dat.hs`, que, sendo carregado (podem fazê-lo noutra janela) e nele executada a função `toTeX`, obter-se-á no ficheiro `out.pdf` o PDF pretendido:

Total: 32

File: Dir.lhs

Attribute	Data
perms	-rw-r--r--
nrlnk	1
owner	jno
group	jno
size	2014
month	Dec
day	1
time	13:19
name	Dir.lhs

File: TeX.lhs

⁵ Haskell a gerar Haskell :-).

Attribute	Data
perms	-rw-r-r-
nrlnk	1
owner	jno
group	jno
size	4796
month	Dec
day	1
time	13:19
name	TeX.lhs

File: j.hs

Attribute	Data
perms	-rw-r-r-
nrlnk	1
owner	jno
group	jno
size	919
month	Dec
day	1
time	14:05
name	j.hs

File: txt.txt

Attribute	Data
perms	-rw-r-r-
nrlnk	1
owner	jno
group	jno
size	0
month	Dec
day	1
time	14:05
name	txt.txt

Note-se que podíamos ter feito tudo num programa só. A vantagem de termos `dat.hs` independente é podermos trabalhar com a informação armazenada em `dat :: Dir` na linha de comandos (de outra forma teríamos que usar *mónades*, um conceito de que devem ter ouvido falar em Programação Funcional mas praticado pouco). Por exemplo, podemos listar os ficheiros ordenados por tamanho correndo

```
sortOn size (files dat)
```

desde que se programe a função

```
sortOn :: (Ord b) => (a -> b) -> [a] -> [a]
```

que ordena listas de valores do tipo *a* de acordo com uma função de inspecção *f* :: *a* -> *b* que converte um qualquer tipo *a* num tipo ordenável *b* (no exemplo acima, essa função é `size :: File -> Int`).

Também se podem seleccionar os ficheiros que satisfaçam um determinado critério, por exemplo correndo

```
select ((>500).size) (files dat)
```

para seleccionar os ficheiros com tamanho superior a 500, desde que se programe a função `select`. E por aí fora...

B Reconhecimento de texto em Haskell

Uma forma de se fazer reconhecimento (vulg. *parsing*) de textos (vulg. *strings*) por forma a os analisarmos é através do uso das chamadas *expressões regulares*. Uma expressão regular é um *padrão* que se pode encontrar numa dada sequência de caracteres. Por exemplo, na palavra *abalar* encontramos o padrão seguinte:

o 'a' repete-se, sendo sempre seguido de uma consoante.

Podemos pois dizer que essa palavra segue o padrão (expressão regular) $(aX)^*$, onde $X = \{b, l, r\}$ e a estrela quer dizer "...e assim sucessivamente". Um exemplo de outra palavra que segue o mesmo padrão é *alabarar* (=denegrir com fogo), havendo muitas outras.

Apesar de as expressões regulares não terem sido leccionadas no decorrer desta disciplina, vimos nas aulas sobre a *shell* alguns exemplos de caracteres especiais, intitulados *wildcards* (* e ?) que nos permitiam realizar algumas acções especiais.

Existem várias bibliotecas em Haskell para facilitar o reconhecimento de cadeias de caracteres utilizando expressões regulares. Uma delas é

```
import Text.Regex
```

onde se define uma expressão regular como:

```
data Regex
```

Deste modo, para criarmos uma expressão regular a partir de uma string basta utilizarmos a função:

```
mkRegex :: String -> Regex
```

Para a comparação da expressão regular com a string utiliza-se a função

```
matchRegex :: Regex -> String -> Maybe [String]
```

É importante notar que o tipo de dados que é devolvido é um `Maybe [String]`. Aconselha-se o uso das funções `isJust` ou `isNothing` (importando a biblioteca `Data.Maybe`) para lidar com o resultado, como por exemplo em

```
isJust (matchRegex (mkRegex "abc") "xaabcd") = True
```

Outra função muito importante é

```
splitRegex :: Regex -> String -> [String]
```

que divide uma string em várias strings, consoante o delimitador (expressão regular) que é fornecido. Veja-se o exemplo:

```
splitRegex (mkRegex ":") "a:b:c:d:e" = ["a","b","c","d","e"]
```

Mais funções desta biblioteca que poderão ser úteis podem ser estudadas no link

```
http://cvs.haskell.org/Hugs/pages/libraries/base/Text-Regex.html.
```

Como exemplo, imaginemos o resultado de invocarmos `ls -lR` numa directoria com duas pastas cada uma com dois ficheiros que se guardou na seguinte string:

```
b=".:\\ntotal 8\\n-rw-rw-r-- 1 carlos carlos      0 2011-12-02 14:14
a.txt\\ndrwxrwxr-x 2 carlos carlos 4096 2011-12-02 14:12
pasta1\\ndrwxrwxr-x 2 carlos carlos 4096 2011-12-02 14:12
pasta2\\n\\n./pasta1:\\ntotal 12\\n-rw-rw-r-- 1 carlos carlos
173 2011-12-02 13:20 txt.txt\\n-rw-rw-r-- 1 carlos carlos
4804 2011-12-02 10:31 x.hs\\n\\n./pasta2:\\ntotal 12\\n-rw-rw-r--
1 carlos carlos 4804 2011-12-02 10:31 a.hs\\n-rw-rw-r-- 1 carlos
carlos 173 2011-12-02 13:20 ex.txt\\n"
```

Para dividirmos a string em várias strings consoante as directorias, temos de a separar aquando dos caracteres "\\.". Basta usarmos o comando

```
splitRegex (mkRegex "\\./") b
```

obtendo-se

```
[".:\\ntotal 8\\n-rw-rw-r-- 1 carlos carlos      0 2011-12-02 14:14
a.txt\\ndrwxrwxr-x 2 carlos carlos 4096 2011-12-02 14:12
pasta1\\ndrwxrwxr-x 2 carlos carlos 4096 2011-12-02 14:12
pasta2\\n\\n",

"pasta1:\\ntotal 12\\n-rw-rw-r-- 1 carlos carlos 173 2011-12-02 13:20
txt.txt\\n-rw-rw-r-- 1 carlos carlos 4804 2011-12-02 10:31 x.hs\\n\\n"

,"pasta2:\\ntotal 12\\n-rw-rw-r-- 1 carlos carlos 4804 2011-12-02 10:31
a.hs\\n-rw-rw-r-- 1 carlos carlos 173 2011-12-02 13:20 ex.txt\\n"]
```

Se agora fosse necessário filtrar quais as directorias que continham ficheiros Haskell, ou seja, com extensão ".hs", bastava correremos o seguinte comando (supondo que o resultado anterior estaria guardado numa variável c):

```
filter (isJust . matchRegex (mkRegex "\\..hs")) c
```

C Instalação das ferramentas

Este anexo contém as instruções de instalação das ferramentas ExifTool e FFMpeg.

- Windows

- ExifTool: o ficheiro <http://www.sno.phy.queensu.ca/~phil/exiftool/exiftool-8.71.zip> deve ser descarregado, descompactado e o ficheiro exiftool(-n).exe deve ser renomeado para exiftool.exe e colocado numa pasta acessível pelo PATH do Windows (por exemplo, podem colocar na directoria Windows/System32).
- FFMpeg: Podem recorrer a duas formas —
 1. Instalar o ImageMagick através do instalador <http://www.imagemagick.org/download/binaries/ImageMagick-6.7.3-9-Q16-windows-static.exe> que já traz o FFMpeg *bundled*;
 2. Descarregar o ficheiro <http://ffmpeg.zeranoe.com/builds/win32/static/ffmpeg-git-b55dd10-win32-static.7z>, descompactar e colocar numa directoria visível pelo PATH. Para descompactarem o ficheiro poderão precisar da ferramenta 7zip (<http://downloads.sourceforge.net/sevenzip/7z920.exe>).

- (OPCIONAL) GnuWin32: Conjunto de ferramentas GNU (ls, cat, wc, grep, less, unzip, tar, sed, awk,...) a funcionar em ambiente Windows. Instalador automático para download em <http://switch.dl.sourceforge.net/project/getgnuwin32/getgnuwin32/0.6.30/GetGnuWin32-0.6.3.exe>

- Ubuntu

- Em Ubuntu, mal se tenta executar uma destas ferramentas, aparece a indicação do comando `sudo apt-get <package>`, necessário para a instalação das mesmas.

- Mac OS X

- ExifTool: o ficheiro <http://www.sno.phy.queensu.ca/~phil/exiftool/ExifTool-8.71.dmg> contém o *package* necessário para instalar o Exif-tool.

- FFmpeg (requer o Xcode disponível na App Store):

```
$ svn checkout svn://svn.ffmpeg.org/ffmpeg/trunk ffmpeg
$ cd ffmpeg
$ ./configure --disable-yasm
$ make
$ sudo make install
```

D Kit Multimédia

Os ficheiros do Kit Multimédia foram escolhidos através de fontes de uso livre e são conteúdos de tipo *Public Domain*, *Royalty Free* e *Creative Commons*, principalmente. Alguns dos sítios a que se recorreu para a sua obtenção foram <http://www.archive.org/>, <http://vimeo.com/>, <http://incompetech.com> e <http://www.8bitpeoples.com>. Outros conteúdos foram obtidos através das páginas dos autores, como é o caso dos álbuns de Nine Inch Nails, Les Triple e Radio99, respectivamente <http://www.nin.com>, <http://www.facebook.com/lestriple> e <http://www.facebook.com/radio99>.

O conteúdo é variado e mostra que mesmo em questões de arte, é possível apreciar, usar, criar e partilhar sem se recorrer a meios *copyrighted*. O mesmo que deveremos sempre fazer no desenvolvimento de software. É pois conveniente tomar consciência do impacto prático que as licenças têm, quer no software a ser usado, quer nos conteúdos.

O vídeo que se segue apresenta um resumo sobre a licença *Creative Commons*, uma das licenças (ou mais correctamente, conjunto de licenças) mais relevantes para a partilha livre de conteúdos artísticos. (Também se encontra incluído no *kit multimédia*, juntamente com uma música do Redd Blood Cells; o álbum não se pode disponibilizar porque o acordo chegado com os White Stripes foi o de removerem o álbum do sítio oficial quando chegasse aos 60000 downloads.)



Figura 2: Explicação das licenças Creative Commons.

Se viram o vídeo e repararam nos exemplos multimédia anteriores, sempre que foi usada uma música de Kevin MacLeod foi dado crédito à sua música. Apesar de o artista Kevin MacLeod (Incompetech) ser o legítimo proprietário e deter o *copyright* da sua música, permite o seu uso músicas desde que lhe seja dado crédito, usando para isso a licença *Creative Commons* ⁶.

A arte que tenha caído em *Public Domain* (como o conteúdo do archive.org) pode ser usada livremente porque já ninguém detém os direitos sobre a obra. Isso permite que a obra seja usada quer na íntegra quer de forma derivada, sem ter de se pagar ou pedir licença aos autores originais. Como exemplos de arte que pertence ao domínio público temos toda e qualquer obra de música clássica⁷ e filmes clássicos como *Nosferatu* do Murnau e *Plan 9 from Outer Space* do Ed Wood.

Outros artistas, como é o caso dos Nine Inch Nails, decidiram lançar algumas das suas obras (*The Slip*) de forma gratuita, seguindo o modelo de negócio *You pay what you want*, onde as pessoas podem contribuir com alguma quantia para o álbum, apesar de ele estar disponível gratuitamente para *download*. Outra banda *mainstream* que seguiu o mesmo conceito foram os Radiohead com o seu álbum *In Rainbows*. Apesar destas músicas serem gratuitas, são *copyrighted* e não podem ser usadas sem autorização expressa dos seus autores.

Do lado oposto temos Tay Zonday com o seu clássico do *Youtube* “Chocolate Rain” que é distribuído através de uma licença *Creative Commons*, onde permite o uso não-derivativo da obra para fins não-comerciais, desde que seja creditado.

Entre as abordagens anteriores temos o caso de um artista que obteve sucesso através do *Youtube*, chamado *melodysheep*, com o seu primeiro single “A Glorious Dawn feat. Carl Sagan”. As músicas seguem esta (<http://creativecommons.org/licenses/by-nc-sa/3.0/>) licença *Creative Commons* e encontram-se disponíveis no sítio do projecto (<http://symphonyofscience.com/>) sendo possível fazer *download* através do *Bandcamp* do artista (<http://melodysheep.bandcamp.com/>) e pagar o que se quiser (“*pay what you want*”) pelo *download* das músicas.

Existe também o caso do filme *Home* (<http://www.homethemovie.org/>) que não tem qualquer *copyright* associado e é feito para ser distribuído sem restrições, de forma a que toda a gente a ele tenha acesso. Outro caso semelhante é o da série de filmes *Zeitgeist* (<http://zeitgeistmovie.com/>) que é *fully copyrighted* mas autoriza o *download* e distribuição de cópias físicas dos seus filmes, apesar de venderem o filme em DVD e não permitirem cópias em sítios de downloads ilegais.

O link para download do kit é:

http://dl.dropbox.com/u/2734257/Media_kit.iso

O kit é fornecido como um ficheiro ISO, que poderão reconhecer como sendo um ficheiro de “imagem de CD/DVD”. Isto permite que em ambientes **nix* possam montar o ficheiro numa pasta, para acederem a ele, sem terem de extrair a informação, usando o comando `mount`.

```
$ mkdir -p /mnt/kit
$ sudo mount -o loop Media_kit.iso /mnt/kit
```

Em ambientes Windows poderão fazer o *extract* do ficheiro, para depois acederem facilmente aos ficheiros para o trabalho, usando por exemplo a ferramenta FLOSS (Free/Libre OpenSource Software) 7zip. <http://www.7-zip.org/>

Podem também usar algo como o Virtual CloneDrive (freeware) para montarem a imagem de CD como uma drive virtual no Windows (<http://static.slysoft.com/SetupVirtualCloneDrive.exe>).

⁶A excepção são as músicas *royalty free*, de que há exemplos no *kit*. Nas músicas *royalty free* como, por exemplo, em maior parte dos *jingles* de anúncios, não é preciso dar crédito aos autores.

⁷Daí que muitos filmes amadores e curtas metragens usem este tipo de música.

Estando munidos desta pequena introdução às variantes de *copyright*, incentiva-se a leitura, para aqueles mais curiosos, dos seguintes sítios:

<http://creativecommons.pt/cms/view/id/28/>

http://en.wikipedia.org/wiki/Public_domain⁸

<http://en.wikipedia.org/wiki/Copyright>

<http://en.wikipedia.org/wiki/Copyleft>⁹

E atenta-se para os paralelos que existe no mundo do software:

http://en.wikipedia.org/wiki/Free_and_open_source_software

<http://www.gnu.org/philosophy/open-source-misses-the-point.html>

http://en.wikipedia.org/wiki/Comparison_of_free_software_licenses

⁸O próprio conteúdo da Wikipedia costuma estar sobre Creative Commons ou GFDL. Ver <http://en.wikipedia.org/wiki/Wikipedia:Copyrights>.

⁹Sem significado legal. Usado principalmente para software; semelhante ao *ShareAlike* da *Creative Commons*.