Instalação de software numa Distribuição Linux

A resolução da presente ficha de trabalho deverá fazer parte integrante de um único relatório que deverá:
- conter a resolução de TODAS as fichas de trabalho até à data de entrega;
- ser submetido em formato pdf, através da plataforma Nónio, dentro do prazo indicado nessa plataforma;
- identificar o aluno e ano letivo nos printscreens realizados do terminal, recorrendo à variável de shell PS1 para introduzir o primeiro e último nome no prompt do terminal.
- seguir o modelo disponível na plataforma Nónio;
- ser realizado por o máximo de um 1 aluno;
- incluir uma análise SWOT;
- ser assinado digitalmente.

Instale na distribuição Linux Fedora **seis dos programas** presentes na seguinte lista de software:

Comunicação:
- Thunderbird (equivalente ao Outlook, multi-plataforma)
- Chrome (browser, multi-plataforma)

Multimédia:
- Mplayer (multimedia player, multi-plataforma)
- Video DownloadHelper (url -> vídeo/audio, multi-plataforma)

Downloads:
- Amule (equivalente ao emule, multi-plataforma)
- qBittorrent (equivalente ao utorrent, Linux e Mac)

Vídeo:
- Avidemux (editor de vídeo, multiplataforma)
- Kdenlive (non linear vídeo editor, Linux and MAC)

Imagem:
- Inkscape (editor de imagem vectorial, multiplataforma)
- GIMP (editor de imagem, multiplataforma)

Áudio:
- Audacity (editor de áudio)
- Clementine (player de áudio, multi-plataforma)

Programming
- Eclipse (IDE, multiplataforma)
- IntelliJ (IDE, multiplataforma)
- Atom (IDE, multiplataforma)

Rede:
- Wireshark (analizador de protocolos, multi-plataforma)
- FileZilla (FTP Client, multi-plataforma)

Diversos:
- Wine (is not an emulator)

1. Descreva o processo utilizado na instalação de cada programa, e a respetiva funcionalidade.
   a. Deverão ser exemplificadas obrigatoriamente as formas de instalação por pacote, quer via CLI como através de GUI.
   b. Será valorizada, mas não obrigatória a instalação a partir do código fonte, e a partir de software em "formato binário".
   c. De notar que apenas uma demonstração de instalação é necessária para cada software. Deverá recorrer-se ao auxílio de printscreens sem, no entanto, se limitar aos mesmos.
2. No caso do Thunderbird deverá apresentar uma descrição não só da sua instalação e funcionalidade, como também da sua configuração com uma conta de email real.
3. Utilizar o programa Wine para instalar quaisquer dois programas com suporte nativo apenas para Windows. Descreva os passos para a sua instalação.
4. Descrever o que são IDEs e apresentar um comparativo (e.g. tabela comparativa) entre os IDEs mais importantes existentes na atualidade.

Anexo

Pretende-se com este trabalho desenvolver competências na instalação de software em ambiente Linux. A instalação de software pode ser realizada de várias formas:

1. recorrendo a **pacotes** de software pré-compilados (e.g. .rpm no Fedora e Red Hat, .deb no Ubuntu e Debian), quer por linha de comando quer recorrendo a um GUI (*Graphical User Interface*);
2. a partir do **código fonte**. Neste caso são normalmente seguidos entre três a quatro passos:
    a. Descomprimir o ficheiro que contém o código fonte
    b. Configurar
    c. Compilar
    d. Instalar
    e. Limpar
3. a partir do **software em binário** na qual é efetuada a execução direta do ficheiro em causa.

Nas duas últimas situações é extremamente importante a leitura do ficheiro README (ou análogo) que usualmente acompanha o software.

1. Instalação através de pacotes de software

Exemplos de formatos de pacotes de software utilizados por várias distribuições de Linux:
- **.rpm** (Redhat Package Management): usado pela maioria das distribuições, como por exemplo a Red Hat, a Suse e a Mandrake;
- **.deb**, utilizado pela distribuição Debian e derivadas como o Ubuntu;
- **tgz or tar.gz** —utilizado pelo Slackware e outros.

Software installed from an RPM package differs from compiling from source in a few ways, but the most important one of all is the software is already compiled for you. Essentially all you are doing is extracting the pre-built binaries and copying them to their pre-selected destination.

yum command (or the recent **dnf** command) does for RPM packages roughly what *apt-get* does for Debian packages. Like *apt-get*, *yum* can download and install packages from a configured repository.

```
[root]# yum install packagename
```

To remove software is just as easy.

```
[root]# yum remove packagename
```

yum does not keep a local copy of your package database by default, so normally there is no need to update it. To install all available security patches and bug fixes, use this command:

```
[root]# yum update
```

You can also explicitly install a package from a local file with:

```
[root]# yum install packagename.rpm
```

Nas versões recentes dos ambientes de desktop Gnome e KDE, basta clicar sobre o ícone de um ficheiro RPM para que o sistema nos pergunte se o queremos instalar.

Adicionalmente, é possível pesquisar e instalar o software de forma gráfica no Fedora recorrendo a: *System->Administration->Add/Remove Software* .

2. Instalação a partir do código fonte

Some software is distributed in "Source form". This means you download a file containing all the source code for the application you want to install, unpack it, and compile it on your system. Compiling is the process of turning the source code into an executable binary. The installation procedure for software that comes in tar.gz and tar.bz2 packages isn't always the same, but usually it's like this:

 # `tar zxvf package.tar.gz` (or `tar jxvf package.tar.bz2`)
 # `cd package`
 # `./configure` (configure the installation)
 # `make` (compile the software to form the executables (binaries))
 # `make install` (install the binaries in the right directories

I - Unpacking

The package containing the source code of the program has a ***tar.gz*** or a tar.bz2 extension. This means that the package is a compressed "*tar* archive" (also known as tarball). When making the package, the source code and the other needed files were piled together in a single tar archive, hence the tar extension. After piling them all together in the *tar* archive, the archive was compressed with *gzip*, hence the *gz* extension.

Some people want to compress the tar archive with *bzip2* instead of *gzip*. In these cases the package has a ***tar.bz2*** extension. You install these packages exactly the same way as *tar.gz* packages, but you use a bit different command when unpacking. It doesn't matter where you put the tarballs you download from the internet but we should create a special directory for downloaded tarballs.

After unpacking, read any documentation you find in the respective directory, like README or INSTALL files, before continuing!

II - Configuring

Usually, but not always (that's why we need to check out the README and INSTALL files) it's done by running the "*configure"* script. We run the script with the following command in the packages's directory:

```
[user]$ ./configure
```

When you run the "*configure"* script, you don't actually compile anything yet. "*configure"* just checks your system and assigns values for system-dependent variables. These values are used for generating a *Makefile*. The *Makefile* in turn is used for generating the actual binary. Note that one of the labels present in the *Makefile* happens to be named ' *install* '.

When you run the "*configure"* script, you'll see a bunch of weird messages scrolling on your screen. This is normal and you shouldn't worry about it. If configure finds an error, it complains about it and exits. However, if everything works like it should, configure doesn't complain about anything, exits, and shuts up.

III - Building

It's finally time to actually build the binary, the executable program, from the source code. This is done by running the "*make*" command:

```
[user]$ make
```

Note that "*make*" needs the *Makefile* for building the program. Otherwise it doesn't know what to do. This is why it's so important to run the *"configure"* script successfully, or generate the *Makefile* some other way.

When you run "*make"*, you'll see again a bunch of strange messages filling your screen. This is also perfectly normal and nothing you should worry about. This step may take some time, depending on how big the program is and how fast your computer is. If you're doing this on an old dementic rig with a snail processor, go grab yourself some coffee. At this point I usually lose my patience completely.

If all goes as it should, your executable is finished and ready to run after "*make"* has done its job. Now, the final step is to install the program.

IV- Installing

Now it's finally time to install the program. When doing this you must be root (use the *su* command). Now when you're root, you can install the program with the "*make install*" command:

```
[root]# make install
```

Again, you'll get some weird messages scrolling on the screen. After it's stopped, congrats: you've installed the software and you're ready to run it! Because in this example we didn't change the behavior of the configure script, the program was installed in the default place. In many cases it's */usr/local/bin*. If */usr/local/bin* (or whatever place your program was installed in) is already in your PATH, you can just run the program by typing its name.

"*make*" uses the file named *Makefile* in the same directory. When we run "*make"* without any parameters, the instruction in the *Makefile* begin executing from the start and as per the rules defined within the *Makefile*. But when you run *"make"* with *install* as the parameter, the "*make"* utility searches for a label named install within the *Makefile*, and executes only that section of the *Makefile*.

The install section happens to be only a part where the executables and other required files created during the last step (i.e. make) are copied into the required final directories on your machine. E.g. the executable that the user runs may be copied to the */usr/local/bin* so that all users are able to run the software. Similarly, all the other files are also copied to the standard directories in Linux. Remember that when you ran "*make",* all the executables were created in the temporary directory where you had unzipped your original tarball. So when you run *make install*, these executables are copied to the final directories.

V - Cleanning

When we ran "*make*" it creates all sorts of files that were needed during the build process but are useless now and are just taking up disk space. This is why we'll want to "*make clean*":

[user]:$ make clean

However, make sure you keep your *Makefile*. It's needed if you later decide to uninstall the program and want to do it as painlessly as possible!

3. Instalação de software em formato binário:

Muitos programas comerciais são distribuídos sob a forma de arquivos, «tar», «tgz» ou «bin», contendo o software em formato binário. O método de instalação destes programas varia muito de caso para caso, pelo que é necessário consultar os ficheiros README ou INSTALL, que costumam ser incluídos no interior dos pacotes de instalação.

Contudo, a maioria das vezes basta executar um ficheiro que faz toda a instalação de forma automática. Este programa por ter o nome de «*setup*», «*install*», ou o próprio nome do programa que muitas vezes se encontra com a extensão .bin. Por exemplo:

```
./GoogleEarthLinux.bin
```