

IIP (E.T.S. Ingeniería Informática)  
Year 2017-2018

*Lab activity 1. Introduction: Linux, Java, and BlueJ*

Introducción a la Informática y a la Programación  
Departamento de Sistemas Informáticos y Computación  
Universitat Politècnica de València



## Contents

<b>1</b>	<b>Objectives and work previous to lab session</b>	<b>2</b>
<b>2</b>	<b>DSIC labs configuration</b>	<b>2</b>
<b>3</b>	<b>The Linux operating system</b>	<b>2</b>
<b>4</b>	<b>Files and directories</b>	<b>6</b>
<b>5</b>	<b>Command features</b>	<b>7</b>
5.1	Command format . . . . .	7
5.2	File and directory access . . . . .	8
5.3	Wildcards . . . . .	9
<b>6</b>	<b>Common commands</b>	<b>9</b>
6.1	Help and information of other commands . . . . .	9
6.2	Directory working . . . . .	10
6.3	File working . . . . .	11
<b>7</b>	<b>Edition, compilation and execution in BlueJ</b>	<b>12</b>
<b>8</b>	<b>Appendixes</b>	<b>17</b>
8.1	Other useful Linux commands . . . . .	17
8.2	Remote access to labs . . . . .	18

## 1 Objectives and work previous to lab session

In this lab activity, the student must get familiar with all the aspects of the working environment that will be used during all the course. Objectives that are aimed at this lab session are:

- Interacting with the Linux operating system in a graphical environment
- Using basic Linux commands
- Organising your workspace by using directories
- Learning how to edit, compile, and execute a Java code in the BlueJ IDE

The student must read in detail the whole document, in order to center in depth in the task to be developed in the lab. Since it is the first session, no previous knowledge is required.

## 2 DSIC labs configuration

Differently to other computing setups, most of data that you will use during your work (e.g., Java programs that you may write) is not stored in the particular computer you work on, but in a virtual system provided by DSIC (cloud).

This means that your vision of the system (the data and applications that you see when working in a computer) is independent of the computer you are working with. Similarly, if you make a remote connection to the DSIC labs <sup>1</sup> (from your home or an ETSInf computer) you will see the same contents than when working locally.

However, each user has available a special folder (directory), named **DiscoW**, that will keep data to be saved permanently. Consequently, data outside that folder could be lost (e.g., when a re-installation of the system occurs). Thus, it is recommended that you use this folder for saving all material that you generate during the lab sessions.

This folder **DiscoW** for each user exists and is common to the two systems (Linux and Windows) present in the DSIC labs computers, so you can access to your data, present in that folder, from any of the two systems. This information will not appear if you connect from other places without accessing the DSIC labs (e.g., from ETSInf labs).

## 3 The Linux operating system

An *Operating System* (abbreviatedly, *O.S.*) is a set of programs that allows an efficient management of the computer components, and that simplifies the achievement of the most relevant computer operations. An O.S. allows, for example, a simple manipulation of storage devices such as hard disks or DVD drives, human interaction devices such as mouse, keyboard, screen, etc., communication and connection (network) devices, and many more.

---

<sup>1</sup>Appendix 8.2 explains how to do it.

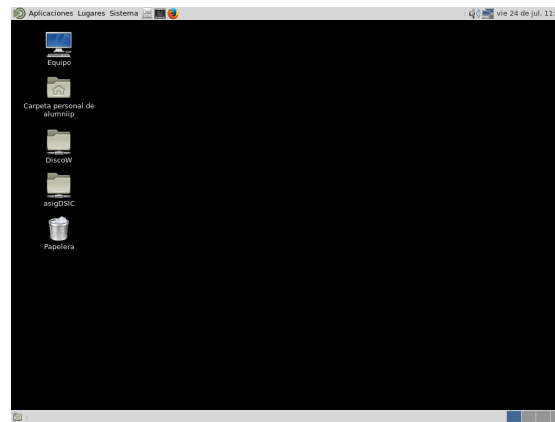


Figure 1: MATE desktop.

Some of the most relevant functions of modern operating systems are:

- Management of the computer resources: storage, input/output, CPU, communications, etc.
- Management of simultaneous users, that allows any user to feel that they are working by themselves (without other users in the same computer)
- Secure access control to the system, in a way that each user can access and manage only those system parts in which he or she is authorised.
- Make easier the interaction with the hardware and provide with graphical and text-based tools that allow the manipulation in the desired form.

In this lab we will use the Linux O.S. Linux is a free, opened (its source code is available and it is possible to modify it) and powerful (Unix-derived) O.S. Linux is widely spread in academia, enterprises, and public services (there is a plan for the Spanish public services to widely use Linux). From the multiple available Linux distributions, in this lab the CentOS 7.2 distribution is installed.

In Linux, like in other O.S. (like Windows or OS X), there are two basic ways of interacting:

- Text-based or *console*, i.e., writing on the keyboard system commands that are executed by the system.
- Graphical-based, using graphical user interfaces (GUI) such as MATE, Gnome, KDE, etc., which allow to use mouse and keyboard to manage the computer by using windows, menus, applications, etc. MATE is the GUI installed in this lab.

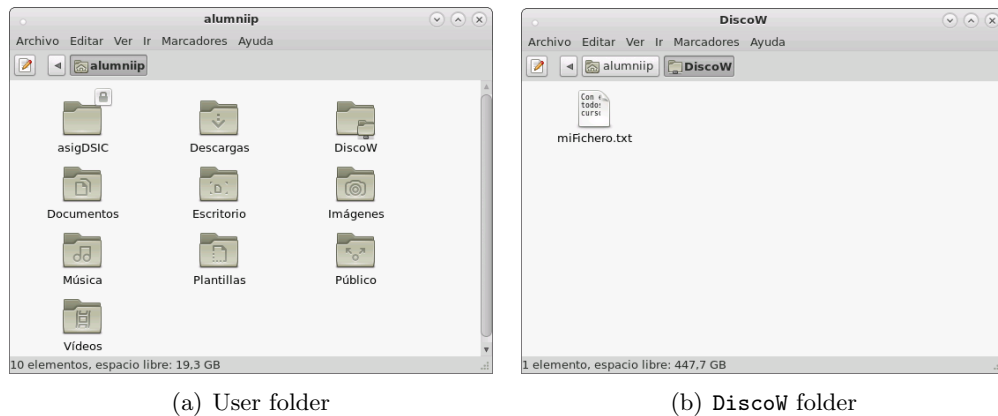


Figure 2: Personal folder for the user, and, in it, DiscoW.

### Activity #1

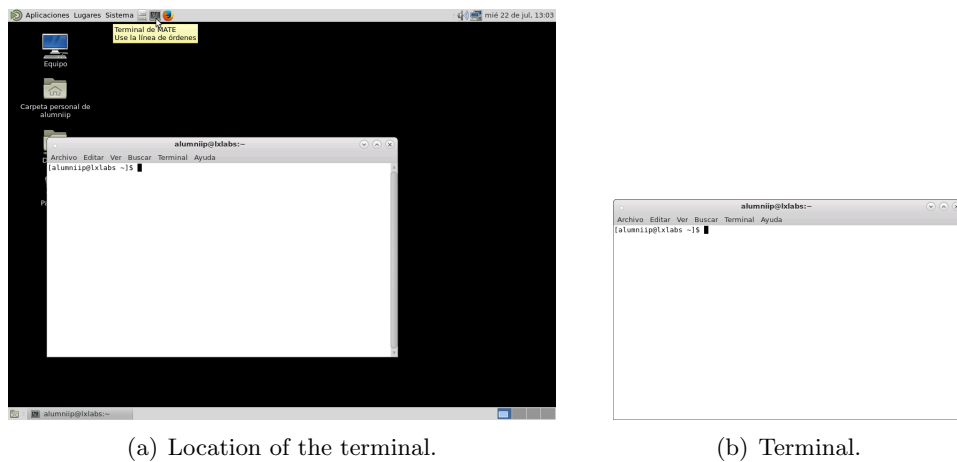
1. Switch on the computer and choose the Windows O.S. option (this is only for validating your password, after that you will have to re-enter with Linux option)
2. After some boot messages, a screen will appear asking for your username (**login**) and your password:
  - Username: the word before @ in your UPV e-mail address (e.g., for **alumniip@inf.upv.es**, it will be **alumniip**).
  - Password: your DNI number (without letter); for non-Spanish students, your foreigner ID number **including the initial letter**

First time you use the system, you will be asked to change the password; after changing it, the system closes session

3. Exit Windows and choose the Linux O.S. option (Linux Estándar); at the end of the process, the systems asks for login and password again; you now must use the new password
4. After opening a session, the screen will show an aspect similar to that in Figure 1, i.e., a graphical desktop where some elements are present: menu bars, Firefox browser icon, and other icons on top; icons for **Equipo** (computer), **Carpeta Personal** (personal folder), **DiscoW**<sup>2</sup>, **asigDSIC**<sup>3</sup>, and **Papelera** (trash) on left center; a bar with the active desktop on bottom; clic the **Aplicaciones** (applications) menu and check the different available applications
5. Make double click in **Carpeta Personal** and look at the window that opens; it contains your personal folder files and folders, with an aspect similar to that of Figure 2(a).

<sup>2</sup>Direct access to your workspace that will be kept for all the sessions in the system

<sup>3</sup>Direct access for a shared disk with folders for several DSIC subjects



(a) Location of the terminal.

(b) Terminal.

Figure 3: Initiating a terminal in MATE.

By using the icons, the menus, or the mouse right button you can manage the contents (copy, paste, move, erase, etc.).

Among the icons you can find the **DiscoW** icon to access that folder, apart from directly accessing from the desktop icon. Both icons can be used for accessing the folder

6. Open the Firefox browser and access to PoliformaT; download into your **DiscoW** folder the file `myFile.txt` that you will have available in the folder **Recursos - Laboratorio - Práctica 1**; check that the file is in your **DiscoW** folder, as in Figure 2(b)

For using command line interaction with the O.S., you need a *terminal* or *console*, that execute a program called *shell*, which interprets text commands.

## Activity #2

1. Start a terminal by using the icon on the top bar (Figure 3(a)); it will appear a window, with an aspect similar to that in Figure 3(b), with a prompt that consists of your username, the identifier of the current connection you are working with, the current directory (folder), and the end character `$`; commands can be written after the `$` symbol.

In general, a command presents the form `command -options arguments`; after writing the command (with its possible options and arguments), it is executed by pressing enter. Before listing the most important commands and their effects, Section 4 describes how data is organised in Linux.

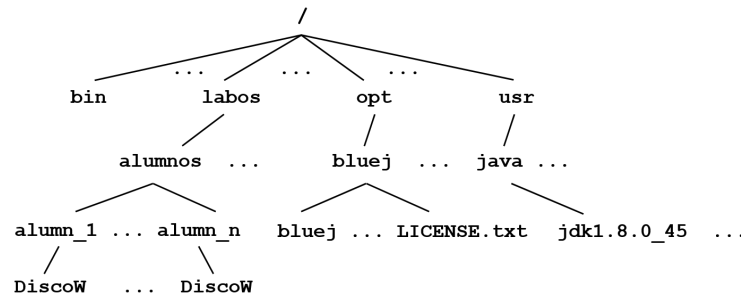


Figure 4: Directory tree

## 4 Files and directories

The basic information element in the system is the *file*. In a simplified manner, a file is a sequence of homogeneous information units that is in the computer. For example, there are character files, numerical files, command files, and, in general, files for all possible information that can be coded.

From the O.S. view, a file has a name that allows to recognise it, and a set of attributes such as its size, its last modification date, its owner, and *access permissions* (i.e., a definition of what can different users do with the file).

The O.S. recognises another element that allows to group inside other informational elements. These elements are called *directories* or *folders*, and, as its name indicates, they contain files and other directories.

There is the *root* directory, which includes hierarchically the rest of directories and files of the system. It is represented by the symbol / (slash).

The system set of directories and files build a hierarchical structure in which some directories contain files as well as directories, which can have a similar internal structure (i.e., they contain other files and directories). This hierarchical structure is usually represented as an inverted tree, with / as root, and with nodes that are directories or files. By similarity, this structure is called *directory tree of the system*. Figure 4 represents a part of the directory tree for our system (notice the *DiscoW* directories present for each user).

Each file or directory of the system is represented in a unique way by the sequence of directories that must be followed in the directory tree from the root / to the file or directory. These steps that must be used to get to the element is usually known as *path* of the file or directory. For example, the sequence: `/opt/bluej/LICENSE.txt` shows the complete path needed to arrive to the file name `LICENSE.txt`.

Any system user is, in a specific moment, in one of the system directories, which is called *working directory*. Being in a working directory means that the operations the user makes on files and directories are, by default, performed on the elements of that working directory. The user can change the current working directory (by using the command `cd`).

When initing a working session, the system automatically situates the user in a

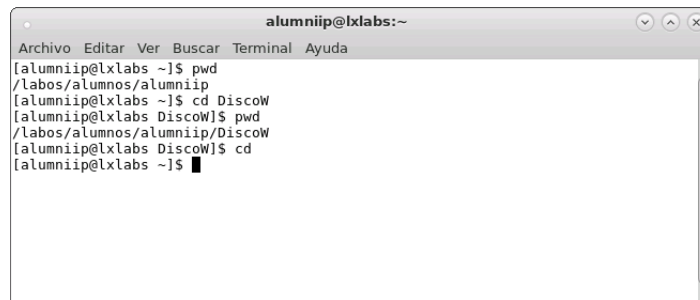
A screenshot of a terminal window titled 'alumniip@lxlabs:~'. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows a sequence of commands and their outputs: 'pwd' returns '/labos/alumnos/alumniip', 'cd DiscoW' changes the directory, and 'cd' without arguments returns to the home directory. The prompt changes from '[alumniip@lxlabs ~]\$' to '[alumniip@lxlabs DiscoW]\$' and back to '[alumniip@lxlabs ~]\$'.

Figure 5: Change of the system prompt.

predetermined directory, the *home* directory. The path of this directory is stored in an environment variable for each user called `$HOME`.

Then, for example, when a user `alumniip` executes the command `pwd` (“print working directory”) after initing a session, the system will show the current directory, i.e., its home directory, `/labos/alumnos/alumniip` for the directory tree in Figure 4.

### Activity #3

1. Check in which directory you are by executing `pwd`
2. Change to directory `DiscoW` by using the command `cd DiscoW`; check with `pwd` that the change was done
3. The command `cd` without arguments returns to directory defined at your `$HOME`; notice that the console prompt changes, showing to you in which directory you are in each moment (see Figure 5)

## 5 Command features

Before describing the most usual commands, it is interesting to know some features related to its use: first, the format of the commands; second, the way the files and directories can be named; and finally how to refer to a set of files simultaneously.

### 5.1 Command format

Shell commands are written as individual words that the system recognises and execute. They may be written along with modifiers (options) that change the behaviour of the command.

**For example:**

- The command `ls` asks the system to list the names of the files that are in the current working directory.

- The command `ls myFile.txt` asks the system to list the file with name `myFile.txt` if it is in the current working directory.
- The command `ls -l -a` asks the system to list all the information (modifier `-l`) associated to the visible files and the invisible files (modifier `-a`) of the current working directory. Modifiers can be grouped (e.g., `ls -la` has the same effect).

**IMPORTANT TIP:** Linux shell can accept partially written commands, that can be automatically completed; thus, after writing the initial letters of a command, pressing the tab key, the shell will try to complete the command (if there is no ambiguity).

**For example:** The command `ls -l my` is not complet if we like to know the features of the file `myFile.txt`. If after writing `ls -l my` the tab key is pressed, then the Linux shell will try to complete the command, that will occur if the only file in the current working directory that starts with `my` is `myFile.txt`, completing the order as `ls -l myFile.txt`.

Apart from that, cursor keys (up and down, or equivalently `CTRL+P` and `CTRL+N`) can be used to retrieve commands previously written (command history).

#### Activity #4

1. Execute the previously listed `ls` orders in your `DiscoW` directory and check if the results are those expected
2. Employ up and down cursor keys to verify that the previously written commands appear after the prompt

## 5.2 File and directory access

In Linux, files and directories can be accessed in an order using two formats:

- **Absolute:** the name of the file or directory is preceded by the path from the root directory. For example, the command `ls /opt/bluej/` asks the system to list the name of the files that are present in the indicated directory.
- **Relative:** in this case, knowing the current working directory, the access to the file or directory uses the sequence of directories from the working directory to the desired resource. For example, if the working directory is `/opt/` and the command is `ls -l bluej/LICENSE.txt` this command asks the system to list all the information (modifier `-l`) associated to the file `LICENSE.txt`, which is in the path indicated by the current working directory and the directories specified in the command (i.e., `/opt/bluej/LICENSE.txt`).

It is important to know some special system references that are used for special directories, which are:

- `..` reference to the parent directory of a given one.
- `.` reference to the current directory.



### Activity #5

1. Use the `ls` command to list all information related to directory `/opt/` by using the `..` reference in order to access to `/opt/` from your `$HOME` directory

## 5.3 Wildcards

Some special characters, known as *wildcards*, are used when writing file and directory names in commands, in order to give them a special meaning. Its use allows to name a set of files with a common syntactic feature, which allows the application of the same command to groups of files or directories, instead to a single one. The most important are:

- `?`: represents any single character, thus, for example, the command `ls myFil?.txt` will list the name of the files that present a single character (any of them) in the place of `?`
- `*`: represents any character sequence, thus, for example, the command `ls my*` lists the names of all the files that begin with the sequence `my`

In any time it is needed to interpret literally any wildcard, it must be written between double quotation marks. For example, the command `ls pract2.*` will obtain the name (if it exists) of the file whose name is exactly `pract2.*`.

## 6 Common commands

In this section some of the most usual shell commands are described. They are grouped attending to their functionality and described in a short way.

### 6.1 Help and information of other commands

The following commands allow to obtain information of other commands, using the command system database:

- `whatis CmdName`. Shows a brief summary of the mission of `CmdName`
- `apropos CmdName`. Shows the database entries in which `CmdName` appears
- `man CmdName`. Manual entry for `CmdName`; you can exit by pressing the `q` key

### Activity #6

1. Check the `ls` command information by using `whatis`, `apropos`, and `man`

## 6.2 Directory working

Keeping your files well organised and structured is really important (specially when you have thousands of files and directories, which is more usual than you think). Thus, you must get properly structured your **DiscoW** directory. Remember that the rest of data you manage in your **\$HOME** directory (including desktop) may dissapear in case of system re-installations.

The following commands will allow you to organise your workspace:

- **pwd** show the path from the root directory to the current working directory
- **ls dir** list files and directories inside **dir**; when **dir** is not specified, is the working directory; with option **-a** shows hidden files (usually system files, their names start with **.**); with option **-l** shows more data of each item (size, owner, access permissions, date of creation,...), with the format:
  - Item type (first letter of first column): **d** means directory, **-** regular file
  - Permissions (rest of letters of first column): three groups of three symbols, groups for owner, group, and other; the symbols mean, in this order, read, modify, and execute; they can be **-** (no permission) or **r** (read permission), **w** (modification permission), and **x** (execution permission)
  - Number of links (second column)
  - Owner (third column)
  - Group (fourth column)
  - Size in bytes (fifth column)
  - Date and hour of last modification (variable size)
  - Name
- **mkdir dir** create an empty directory following the specifications of **dir**
- **du dir** shows the space occupied by directory **dir** and all its internal files and directories; option **-c** shows a total; option **-h** offers a letter associated to the size (**M** for megabytes, **G** for gigabytes,...); option **--exclude=dirName** excludes directory **dirName** from size calculations
- **cd dir** change from the current directory to the reference directory **dir**; if **dir** is omitted, changes to **\$HOME**; if **dir** is **..**, changes to the parent of the working directory
- **rmdir dir** remove **dir** only if it is empty
- **tree dir** shows **dir** in the form of a tree

**Activity #7**

1. Create in your `DiscoW` directory the `testIIP` directory, and inside of this, create the subdirectory `labact1`
2. Change to `labact1` and, by using an only command, create the subdirectory `labact1b` inside `testIIP`
3. Change to your `DiscoW` directory and, with an only command, create subdirectory `test` inside `testIIP`
4. List the contents of the `testIIP` directory, including hidden files; result must be similar to:

```
total 20
drwxr-xr-x  5 alumniip alumnos 4096 sep  4 15:47 .
drwx----- 33 alumniip alumnos 4096 sep  4 15:45 ..
drwxr-xr-x  2 alumniip alumnos 4096 sep  4 15:45 labact1
drwxr-xr-x  2 alumniip alumnos 4096 sep  4 15:47 labact1b
drwxr-xr-x  2 alumniip alumnos 4096 sep  4 15:47 test
```

5. Show the total size, in megabytes, of your `DiscoW` directory.

**6.3 File working**

The following commands are used for file manipulation.

- `cp OrgFile DstFile` copies `OrgFile` to `DstFile`; if `DstFile` is a directory name, copies `OrgFile` inside `DstFile`
- `mv OrgFileDir DstFileDir` change name of file or directory `OrgFileDir` to `DstFileDir`; if `DstFileDir` is an existing directory, moves file `OrgFileDir` into directory `DstFileDir`
- `rm FileName` removes file `FileName`; with option `-i` asks user confirmation; with option `-r`, when `FileName` is a directory, removes the directory and all its contents
- `cat TextFileName` shows the contents of `TextFileName`; several files can be passed and they will be shown concatenated; if any file does not exist, it produces an error
- `more TextFileName` shows the contents of `TextFileName` page by page, from the first one to the last one; pressing space key makes it advance one page, and pressing `p` key makes it go back one page; exit is achieved by pressing `q`
- `less TextFileName` is a more powerful version of `more`, that allows to show the contents of `TextFileName` page by page but allowing forward and backward movements, pattern search, etc.; it is only available in Linux

- `wc TextFileName` shows the total number of lines, words, and characters of `TextFileName`; options `-l`, `-w`, and `-c` allow to obtain only the number of lines, words, or characters respectively

### Activity #8

1. From your `DiscoW` directory, check that the directory structure for that directory is that shown in Figure 6(a)
2. Copy the file `myFile.txt` to directory `testIIP/labact1`; check the contents of that directory
3. Erase the `myFile.txt` file from your `DiscoW`
4. Change to `testIIP/labact1` and create a copy of the file `myFile.txt` called `myFile2.txt`; check the contents of the directory and, by using the `cat` command, show the contents of the files `myFile.txt` and `myFile2.txt`
5. With an only command, move the `myFile2.txt` file into directory `testIIP/test`; check the contents of the subdirectories `labact1` and `test`
6. Change to `testIIP/test`; change the name of the file `myFile2.txt` to `myF.txt`; check the contents of the directory and show the contents of `myF.txt`
7. Show the contents of the file `myF.txt` by using the commands `more` and `less`
8. Show the number of lines, words, and characters of the file `myF.txt`
9. Change to `testIIP` and move directory `test` into directory `labact1`; check that the directory tree in your `DiscoW` is that shown in Figure 6(b)

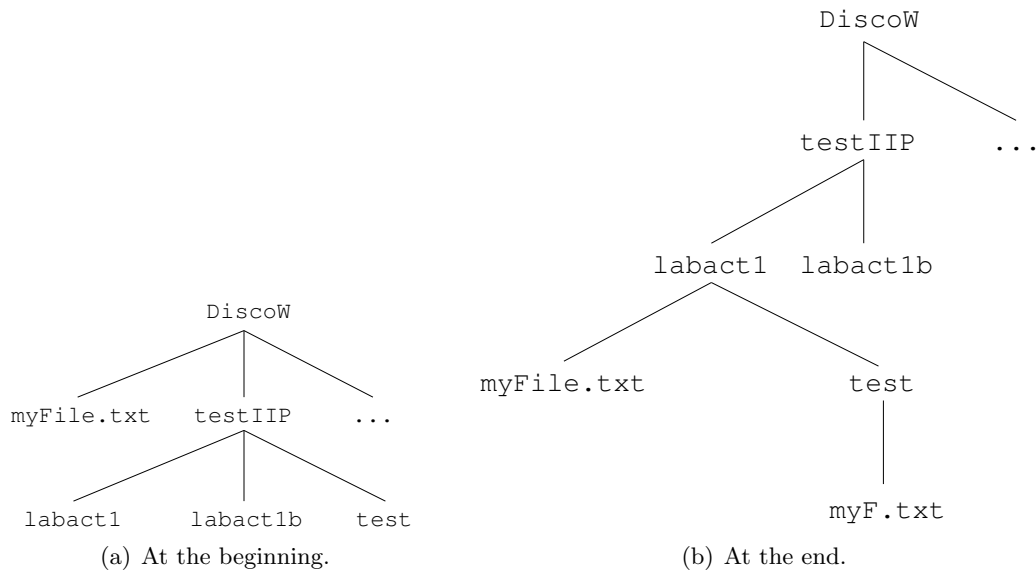
## 7 Edition, compilation and execution in BlueJ

*BlueJ* is a Java programming IDE, created initially by the *BlueJ* team of the Monash University, Melbourne (Australia), and it is currently maintained by research teams from Kent University, Canterbury, and King's College, London (United Kingdom), and which was created with the purpose of being used in introductory courses to the Java language.

Using a quite simple graphical interface, *BlueJ* allows to develop any Java application. In *BlueJ*, the directory that keeps all the files of the Java application is called **project**. It contains the source codes (with `.java` extension) and the bytecodes (with `.class` extension).

Although the *BlueJ* complete functionalities will be detailed in the second lab activity, and you will employ it in the rest of the lab sessions, now it is described how to edit, compile, and execute a Java program.

For executing *BlueJ* you can access to the menu `Aplicaciones - Programacion - BlueJ 3.1.7` or write on the command line:

Figure 6: Directory tree for your `$HOME` directory.

`bluej &`

After calling to *BlueJ*, it appears automatically in a window, the *BlueJ main window* (Figure 7(a)). Notice that its central zone is empty.

In the upper part of the main window there is a menu that allows to access to some operations: on the project (**Project**), edition operations (**Edit**), *BlueJ* tools (**Tools**), viewing (**View**), and help (**Help**).

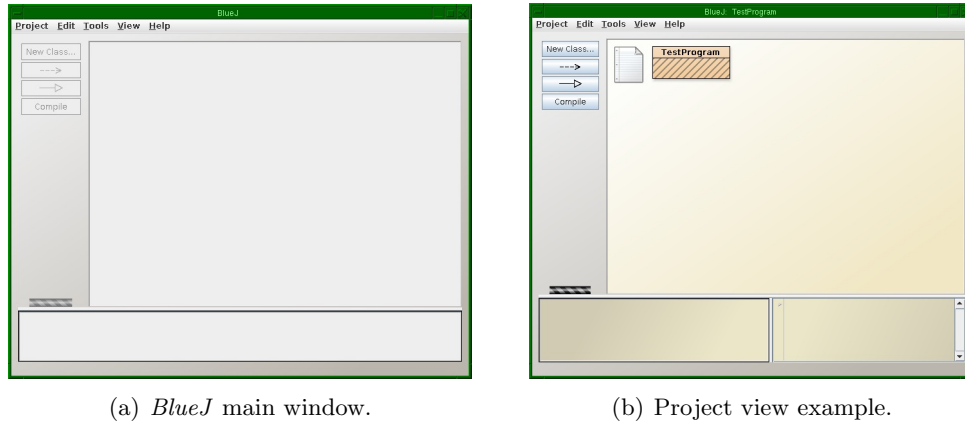
To create a new *BlueJ* project, select the option **Project - New** from the menu and give a name to it.

For creating a class in a project you must click on the button **New Class...** and a name and type (in our examples, we will always choose **Class**). In *BlueJ* main window appears the icon of the created class (Figure 7(b)), that appears with strips since it is not compiled yet.

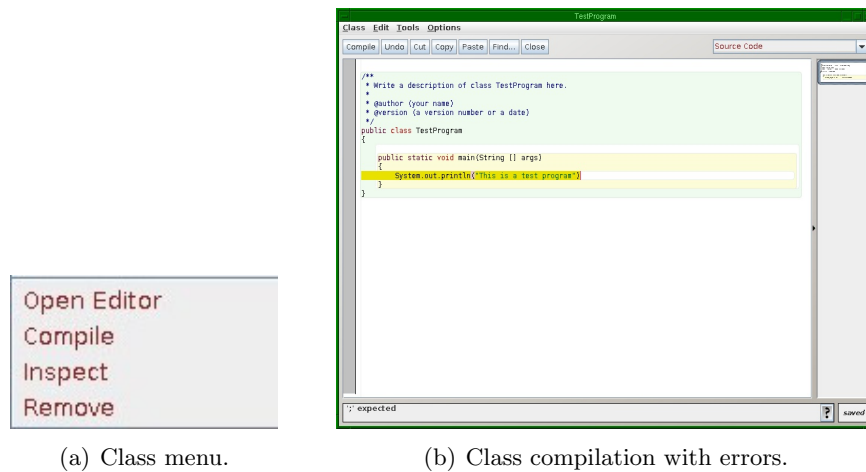
After creating the class, when its icon is pointed and clicked with the mouse right button, the *class menu* (the operations that can be used on the class) is shown (Figure 8(a)). For editing a class the option **Open Editor** from the menu class can be selected. As an alternative, double click on the class icon can be used.

With respect to how compiling a class, there are different ways: from the editor, from the class menu, or from the **Tools** menu of *BlueJ*. If the compiler detects any error, the corresponding line appears shadowed, and a message error will appear in the *information zone* as well (bottom part of the screen) (Figure 8(b)). In this case, when the button with the symbol “?” (right part of the information zone) is clicked, more information can be obtained about the detected error.

Apart from compilation errors, code must be written following an optional (but recommended) style that can be checked from the **Tools - Checkstyle** option in the

(a) *BlueJ* main window.

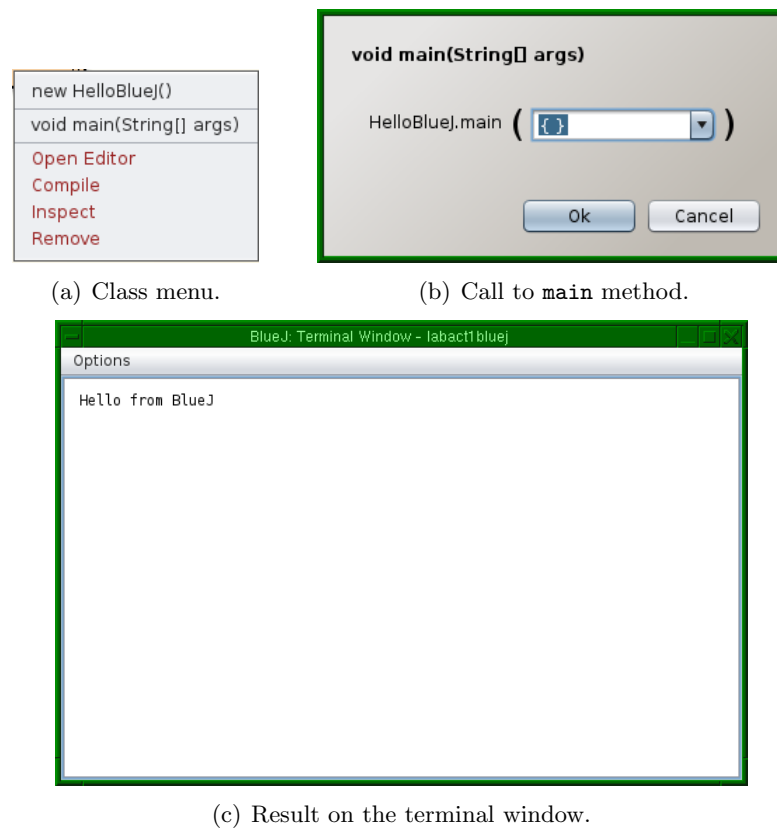
(b) Project view example.

Figure 7: *BlueJ* windows.

(a) Class menu.

(b) Class compilation with errors.

Figure 8: HelloBlueJ class compilation.

Figure 9: Execution of the class `HelloBlueJ`.

project view. When choosing this option, a new window will open with the classes with style errors highlighted in yellow. When clicking the class name, warning messages will appear signaling the parts of the code that do not accomplish the predefined code style that would be used during the subject.

As was said before, when a class is clicked by using the mouse right button, the class menu appears. From the operations that shows this menu, the call to the `main` method must be highlighted, since *allows to execute a class from its menu* (Figure 9(a)).

In Figure 9(b), the main method call window for the class `HelloBlueJ` is shown. When the program asks data from the keyboard or shows results on the screen, a text terminal appears automatically (Figure 9(c)). If it does not appear, select the option `View - Show terminal` from the menu.

### Activity #9

1. From `testIIP/labact1`, call to *BlueJ* without parameters and create the project (which is actually a directory) `labact1bluej`, by using the option `Project - New`
2. Create the class `HelloBlueJ`, and open the editor to write the following code:

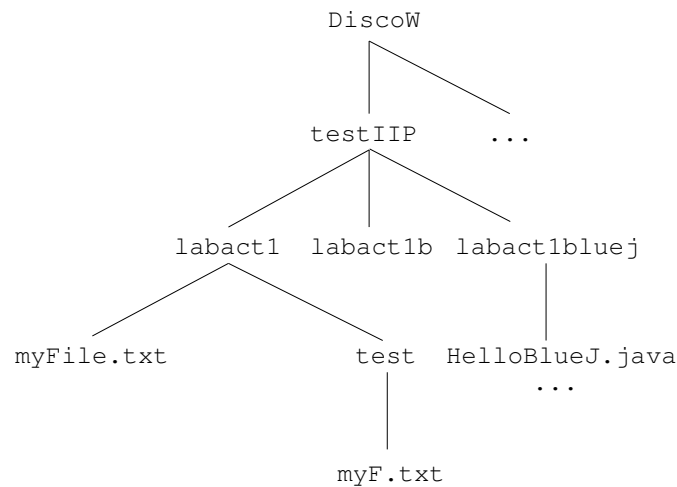


Figure 10: Directory tree in your DiscoW after Activity #9.

```

/** Class HelloBlueJ */
public class HelloBlueJ {
    public static void main(String[] args) {
        System.out.println("Hello from BlueJ");
    }
}

```

3. Compile the class, correct the compilation errors, and check what happens in the central part of the *BlueJ* main window
4. Check the style and correct all possible style mistakes
5. Execute the `main` method of the `HelloBlueJ` class and check that the result is that shown in the Figure 3(b); you can save that result into a text file with the option **Options - Save to** from the terminal window menu
6. Check that directory tree has now the aspect in Figure 10.

**IMPORTANT NOTICE:** when finishing the activity, **DO NOT SWITCH OFF THE COMPUTER**; choose **Cerrar sesión** from the **Sistema** menu, and choose **Cerrar sesión** in the window that appears; after closing MATE, the login screen will appear again.



## 8 Appendixes

### 8.1 Other useful Linux commands

#### Input/output redirection

Most commands accept data from keyboard (standard input), process that data, and show results on the screen (standard output). The following techniques allow to redirect input/output in order to obtain input from a file or record output into a file.

- `Cmd > DstFile` sends to file `DstFile` the output of `Cmd`; if `DstFile` does not exist, it is created; otherwise, its contents are overwritten
- `Cmd >> DstFile` like the previous one but adding the output at the end of `DstFile`
- `Cmd 2> DstFile` sends to file `DstFile` the error output of `Cmd`
- `Cmd < OrgFile` take as standard input for `Cmd` the contents of the file `OrgFile`
- `Cmd1 | Cmd2` sends the output of `Cmd1` as input for `Cmd2`
- `tee DstFile1 ... DstFilen` copies its standard input to its standard output and to all the files that are passed as arguments; by combining `tee` and `|`, the output of a command can be sent to a file and, at the same time, be shown on the screen

#### Access control, security and privileges

The following commands allow to change system password and permissions of a file (who can access and what s/he can do on the file).

- `passwd` changes password
- `chmod mask name` changes the access permissions for file or directory `name` following the pattern given by `mask`

#### Basic system management

Other commands employed for managing the system are the following.

- `who` shows the name of the current system users
- `whoami` shows the name of the user that executed the command
- `&` when written after a command, the command is executed concurrently in background, without blocking the terminal while it is being executed
- `ps` lists active processes
- `kill -9 procid` terminates immediately the process with id `procid`
- `env` shows user environment and it allows its modification
- `exit` ends the current session (closes terminal)

## 8.2 Remote access to labs

DSIC labs can be remotely accessed (for both Linux and Windows instalations) by using remote desktops utilities. By using this, you can work from any computer with Internet access as is you were in the lab.

In the URL <http://www.upv.es/entidades/DSIC/index-en.html>, link “Remote desktops (EVIR)”, you have the description of the programs and basic parameters you must use to do so from any of the supported systems (Linux, MacOSX, and Windows).

Its use is easy, although is worth having a fast Internet connection (current common ADSL connections are enough). Moreover, when starting the remote session you must identify yourself with the username and password you employ in the DSIC labs.

For example, you can access from a Windows system to a Linux work session with the same desktop and resources that you have accessible from a lab session. The same can be done with a remote Windows session, from Mac, Windows or Linux with the proper support.

DSIC recommends using these resources and provides a basic guide for its installation and use.