Language Processors Lab Week 1 (Revisited)

These sheets introduce you to the Linux/Unix Java programming environment to be used on the module by getting you to compile and run a program. It is also a gentle introduction to Language Processors since you will compile and execute your first intepreter: The Straightline Programming Language interpreter

You will be expected to be able to copy and manipulate program files and directories or web locations named in lab sheets, and compile and run the programs, and then to modify them and compile and run them. If you are not familiar with the Linux labs you should also get a copy of any *Computing Services* documentation available.

The Linux Environment

Working with the KDE windowing system under Linux

Each computer is attached by a network to several file systems. Under Linux, the fact that there are multiple file systems is something you don't normally have to think about. All the available systems just appear as different directories (folders) in a large'tree' of directories. The tree is upside-down with its root at the top. The Root Directory is called "`/`". We will use the words 'directory' and 'folder' to mean the same thing.

The Linux labs use Suse Linux and the KDE windowing system as its main user interface. You will notice that KDE has a Windows-like **taskbar** (the control panel) at the bottom of the screen with a button with the Suse logo (K Menu) on the bottom-left. As on Windows, clicking on this reveals a menu that launches applications and utilities.

Your CSD home directory and the KDE File Manager

When logged in as a Linux user you have a *home directory*. This is a place in the file system which belongs to you: the place where you can create, modify and delete files and folders. When you start the file manager (the button labelled with a small house icon on the control panel is the easiest way to start it), your home directory is the one it will display. You can also get the file manager to take you to your home directory at any time simply by clicking on the home icon on the task bar at the top. Use the file manager to go to your home directory now. Does it contain any files? Does it contain any hidden files? (Hint: look on the View menu.)

Unix sessions, Terminals, and your directories

A Unix session is made up of a series of commands typed and executed by the user from the command line at a *Unix terminal* or *Unix shell* window. A Unix shell window can be fired up from the K menu or from one of the icons on the taskbar: the one with the terminal screen on it, marked with **Konsole**.

Fire up a shell window. The first question you may want to ask is 'Where am I?'. You can answer this question by typing:

```
pwd
```

The `pwd` command displays the name of the *current directory*, that is, the name of the directory where you are currently located. By default you will always start a Unix session from your home directory. Therefore, the first `pwd` command should display a directory name starting with '`/`' (the root directory) and ending with your user name. In my case, I got the following:

```
/u4.bristol/s00/sbbc287
```

Now type:

```
ls
```

The `ls` command displays the contents of the current directory - the previos command shoul've listed the contents of your CSD home directory. You can also display the contents of any directory by passing a directory name as argument to the `ls` command. For example, in order to see the contents of the `KDesktop` directory simply type:

```
ls KDesktop
```

Besides arguments, Unix commands can also accept special options listed after a the '-' (minus) symbol. For example, these are valid `ls` options:

`ls -l`	Lists permissions, ownership and last update info
`ls -a`	Lists hidden files as well
`ls -al KDesktop`	Combined options

You can see a brief explanation of almost all Unix command by using the `man` and `info` commands/ For example, type `man ls` and `info ls`.

Creating a directory

In this section you will learn how to create new directories and move through the directory tree. The command `cd dirname` changes the current directory to the directory name specified as argument i.e. `dirname`. First, make sure the current directory is your home directory. You can always move to your home directory by typing the following:

```
cd
```

The `cd` command with no arguments will always get you to your home directory. Now, execute the following series of commands:

`ls`	Check `KDesktop` is one of the names listed
`cd KDesktop`	The current directory should be now be `KDesktop`
`pwd`	Check your current directory
`ls`	Inspect the contents of `KDesktop`
`cd ..`	Move to the parent directory (one directory up)
`pwd`	Check you are back to your home directory

New directories can be created with the `mkdir` command. We would like to create the `LanguageProcessors` directory - you can pick any name you want - in order to store all our lab work. Do the following:

`mkdir LanguageProcessors`	Creates new directory `LanguageProcessors`
`cd LanguageProcessors`	You should be in `LanguageProcessors now`

Java environment and programming

The Java environment we will use is Suns JDK (Java Development Kit, now also known as the SDK), which can be considered to be the reference Java environment, and provides (amongst other things) a compiler, interpreter and debugger. Theseare not integrated into a closed environment, so you need to use at least the Uni file system as above, and a Unix editor to be able to create and modify programs (see below), and learn the compiler and other commands.

Before we continue, make sure your current directory is `LanguageProcessors`[1].

Compiling Java programs on Unix

By default, the CSD labs do not have the `java` environment loaded. In order to use Java in a shell window you must add module `java`, which gives access to the Java JDK (also known as SDK) and some local Java libraries:

[1]Or the directory you have chosen for your Language Processors work

```
module add java
```

IMPORTANT: Always load the `java` module when working with the CSD Unix labs.

Next, we need to get the source Java files that make up the Straightline interpreter. The files have been packaged into an archive file named `straightline.tar` by the Unix utility program `tar`. Do the following:

- Download the `straightline.tar` file from the module's CitySpace directory `Week1`[2] or from the following URL: http://www.soi.city.ac.uk/ sbbc287/straightline.tar.
- Make sure `straightline.tar` has been copied to your `LanguageProcessors` directory.
- Unarchive the files with the following command:

  ```
  tar -xvf straightline.tar
  ```

  This creates the `straightline` directory and copies all related `.java` files into it.
- Change to the `straightline` directory (`cd straightline`) and list its contents (`ls`). You should get the following:

  ```
  Interpreter.java        ParserTokenManager.java TokenMgrError.java
  ParseException.java     SimpleCharStream.java   straightlineAST.java
  Parser.java             Table.java
  ParserConstants.java    Token.java
  ```

  The Straightline interpreter (scanner and parser included) is made up of 10 `.java` files.

The Java compilation command is `javac`, which will compile single or multiple `.java` files and generate `.class` (Java bytecode) files. In order to compile the Straightline interpreter execute the following command:

```
javac *.java
```

which will compile all Java source files in the current directory (remember you must be in the `strightline` directory. *Your first compilation will take a long time; subsequent ones take much less time.*

The `javac` command generates a new `class` file per declared class. Succesful execution of the command above should generate 23 new `.class` files.

Running Java programs on Unix

Java programs are executed with the `java command`:

```
java  MainClass
```

where `MainClass` is any class which contains the `public static void main(String[] args*)` method. In our example, the `main` method is part of the `Interpreter` class. You can quicly check this by invoking the `cat Interpreter.java` or `more Interpreter.java` commands - with the latter, press the spacebar in order to see the rest of the file. Thus, the command

```
java Interpreter
```

will execute the Straightline interpreter.

[2] You may get straightline1.tar due to CitySpace's wickedness

The Straightline Interpreter

When executed, the Straightline interpreter display a couple of greeting lines, followed by the interpreter's prompt `Enter Program >` as shown below:

```
Interpreter of one-line programs.
End input with <Enter>. Terminate interpreter with empty program.
Enter Program > _
```

Recall that the interpreter implements the Strightline programming language, which syntax is defined by the following grammar:

Stm	→	Stm ; Stm | id := Exp | `print`(ExpList)
Exp	→	id | num | Exp Binop Exp | (Stm , Exp)
ExpList	→	Exp , ExpList | Exp
Binop	→	+ | - | * | /

The file `Interpreter.java` contains the entry point to the program: an infinite loop that reads input from the user, parses it and prints out the result. The parser has been implemented using Javacc (to be learned in the following weeks); it performs lexical analysis and creates an instance of the `Stm` class. The file `straightlineAST.java` contains all the classes that implement the programming language's abstract syntax trees and the implementations of interp. Inspect both files and understand the code that implement the interpreter.

Try the following examples:

`print 5`	Parse error - correct it
`print(1,1+2,3*5)`	This should be ok
`print(1,1+2), print(3*5+5)`	Another parse error. What's wrong?
`x = 12; print(10 + x)`	
`x := 12; print(10 * y + x)`	A semantic analysis should've detected this
`x := 5; print(3, (x := x * 10, x+2),500)`	try more of these

Logging Out

Always remember to log out when you have finished working at a machine. To logout from a Linux machine, press the button (K button?) at the left of the control panel at the bottom of the screen, and select Logout from the menu.

If you leave a machine without logging out you are putting your own work at risk.