Operating Systems

Practical 4

ThreadMentor Set-up

Dr. Kevin Farrell
February 2025

This page intentionally left blank

Table of Contents

Legal

1. Aims and Objectives

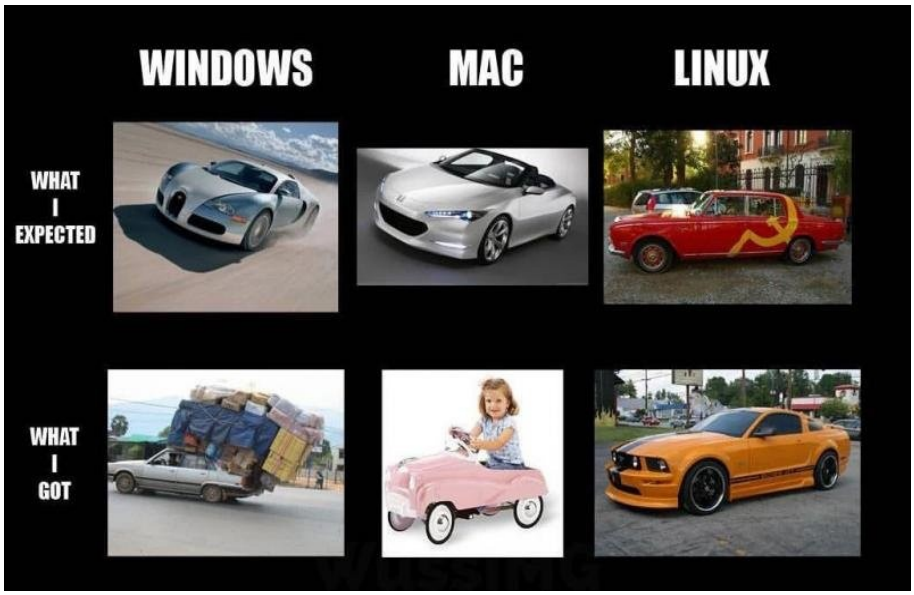
The aims and objectives of this practical are as follows:

1. To set up the ThreadMentor software on your system.
2. To download the code files for your chosen solution to the Dining Philosophers Problem and get it working with ThreadMentor.

2. Summary

The following is a summary of the steps for this practical. The details are on the following pages.

3. To download and extract the ThreadMentor software to the correct location on your Linux system.
4. To perform some basic configuration of your system so that ThreadMentor can be used.
5. To test your ThreadMentor installation using the code for your chosen solution to the Dining Philosophers Problem (obtained from the ThreadMentor e-book website)
6. To begin to learn, with the aid of ThreadMentor, how the code for your chosen solution to the Dining Philosophers Problem operates.
7. To learn about the various features of ThreadMentor.



3. Getting Started

3.1. Boot your **Mageia 8 i586 (32-Bit)** Linux Virtual Machine (VM), and ensure that you have Internet access on it. Ask for assistance if necessary!

3.2. In VirtualBox, use Right-CTRL-F to enter full-screen mode. Right-CTRL is the Control key on the right-hand side of the keyboard. It is referred to as the *host* key. In VMware, click on the four-arrow icon near the top of the window to enter full-screen mode. **Note**: the host key sequence on VMware is CTRL-ALT. This allows you to return to Windows.

3.3. Log in using the following credentials:

- User: **student**
- Password: **password**

3.4. Run the Firefox browser from within your VM to access BrightSpace, and open this handout.

4. Installing Dependency Packages

4.1. Before we set up the ThreadMentor software, we need to install some additional software libraries needed by it.

4.2. On the Operating Systems BrightSpace page, under the **Project** unit, find the sub-unit called **ThreadMentor Source-Code Files**. Download the following files from this folder by clicking each one in turn and selecting "**Download**":

```
ThreadMentor-fedora4.tgz

libglib1.2-1.2.10-24mdv2011.0.i586.rpm

libgtkPLUS1.2-1.2.10-51mdv2011.0.i586.rpm
```

4.3. Using the Internet, look up what the letters “**rpm**” stand for, and write an explanation in your notebook.

4.4. Now, from a terminal window (Konsole), log in as **`root`**:

```
$ su -

Password: qwerty
```

4.5. Note how the command prompt symbol has changed to a # symbol, indicating that you are now logged in as **`root`**.

4.6. Change to the student’s **`Downloads`** directory (note the **`~`** symbol) by typing the command below. **Note**: there is no space between the tilde “**~**” and **`student`**:

```
# cd ~student/Downloads
```

4.7. Type the following to install the first software library you downloaded from BrightSpace (note the **“`./`”** preceding the library name):

```
# urpmi ./libglib1.2-1.2.10-24mdv2011.0.i586.rpm
```

4.8. During this procedure, additional packages will be automatically selected for installation. Answer “y” for “yes”, when prompted. If you get errors about bad signatures, just ignore these. Answer “y” for “yes”, if prompted.

4.9. Now type the following to install the second software library you downloaded from BrightSpace (again, note the **“`./`”** preceding the library name):

```
# urpmi ./libgtkPLUS1.2-1.2.10-51mdv2011.0.i586.rpm
```

4.10. As before, during this procedure, additional packages may be automatically selected for installation. Answer “y” for “yes”, when prompted. If you get errors about bad signatures, just ignore these. Answer “y” for “yes”, if prompted.

4.11. Now logout as **`root`**, by typing the following. You should get the “**`$`**” prompt back indicating that you are a back as **ordinary** user again:

```
# exit

$
```

5. Installing ThreadMentor

5.1. Earlier in this practical, you downloaded the **`ThreadMentor-fedora4.tgz`** file to your **`Downloads`** directory. This file is a *gzipped* (compressed) *tarball* containing all of the ThreadMentor files needed to set up ThreadMentor for your project. In the instructions below, you are going to unzip and extract this *gzipped tarball*[3].

5.2. Using the Internet, look up what a *tarball* is, and write an explanation in your notebook.

5.3. In the top-level of ***your*** home[4] directory, use the command below to create a new directory in which to store ThreadMentor files and associated directories. **Reminder**: Linux file- and directory-names are case sensitive. So, you *must* use lowercase and uppercase letters ***exactly*** as written in these instructions:

```
$ cd

$ mkdir os
```

3 Gzipped tarballs can have extensions: “.tgz” or “.tar.gz”
4 Your home directory is **`/home/student`**, whereas ***the*** home directory is **`/home`**

5.4. Using the command-line ***not*** the GUI, copy the **`ThreadMentor-fedora4.tgz`** file from the **`Downloads`** directory into your newly-created **`os`** directory. If you can't remember how to do this, review the work you did on Linux commands in Practical 02. Write in your notebook, the command(s) you typed for this step. If you are stuck, ask for help from your Lecturer.

5.5. Then, using the command-line, extract the tarball as follows:

```
$ cd os

$ gunzip ThreadMentor-fedora4.tgz

$ tar -xvf ThreadMentor-fedora4.tar
```

5.6. The last two commands *unzip* the gzipped **`ThreadMentor-fedora4.tgz`** tarball, and then *extract* the contents of the tarball. **Note**: For future reference, there is a shorter way to perform these as a *single* command by typing the following (Don't do this!):

```
$ tar -zxvf ThreadMentor-fedora4.tgz
```

5.7. The above commands automatically create a new sub-directory, inside the **`os`** directory, called **`ThreadMentor`** into which all of the files in the tarball are extracted.

5.8. You can now safely remove the ThreadMentor tarball from your **`os`** directory just to be tidy (you should still have a copy in the **`Downloads`** directory). Note: the "**`-f`**" option here *forcibly* removes the file; i.e. without asking you "Yes/No?":

```
$ rm -f ThreadMentor-fedora4.tar
```

5.9. Examine the contents of the newly created **`ThreadMentor`** sub-directory The purpose of this is simply to *look* at the contents, *not* to do anything else:

```
$ cd ThreadMentor

$ ls

$ cd bin/

$ ls
```

5.10. Note the executable file "**visual**" in the above directory. This will be run automatically by ThreadMentor to display a visualisation of the running threads when you run your Dining Philosophers Problem solution. In this case, your program tries to invoke the necessary ThreadMentor visualisation tool, **visual**, stored in **/home/student/os/ThreadMentor/bin** directory.

```
$ cd ../include/

$ ls

$ cd ../Visual/

$ ls

$ cd ../NoVisual/

$ ls
```

5.11. When you're finished looking at the contents, return to *your* home directory:

```
$ cd
```

5.12. **Note**: you could have instead typed:

```
$ cd  ~
```

5.13. Here the "**~**" is a short way of typing **/home/student**

6. Final Set-up: Setting the PATH environment variable

6.1. Each time you type a command, let's say **mycommand**, the shell, **bash** searches in the directories listed in an *environment variable* called **PATH** to try to find the executable file named **mycommand**. If it can't be found, **bash** returns (i.e. writes to the console) an error "**bash: mycommand: command not found**".

6.2. Using the Internet, look up what an *environment variable* is, and write an explanation in your notebook.

6.3. For ThreadMentor to work, **bash** needs to know the location of the **visual** executable file that we mentioned earlier. To do this, in the steps that follow, we are going to set the **PATH** to include the path to (location of) this file, which you will recall is stored in the **bin** subdirectory of the **ThreadMentor** directory – see point 5.10. above.

6.4. First, type the command below, to print to the **console**, the value of your *existing* **PATH** environment variable. **Note**: to obtain the "**value***"* of an environment variable, one has to precede it by a "**$**" sign. Hence, the **$PATH** here is important – you must type that "**$**" sign:

```
$ echo $PATH
```

6.5. Write, in your notebook, the output you obtained. Note that there is no mention in the output of the ThreadMentor **bin** directory. So, at this point, **bash** will not know how to find the **visual** executable file that we mentioned above. We are about to fix that!

6.6. Your Current Working Directory (CWD) should already be *your* home directory. Edit the **.bashrc** file in it, by typing the command below. Note the "." at the start of **.bashrc**:

```
$ emacs .bashrc
```

6.7. At the end of the file (i.e. below the existing text), insert the following two lines:

```
PATH=$PATH:/home/student/os/ThreadMentor/bin

export PATH
```

6.8. Use **C-x C-s** (where **C = CTRL**) to save the file, and **C-x C-c** to exit **emacs**.

6.9. In the **.bashrc** file, you are adding a new directory to the existing **PATH**, namely, the **/home/student/os/ThreadMentor/bin** directory.

6.10. The **.bashrc** file executes every time you open a **konsole** terminal window. So, close your **konsole** terminal window, and run **konsole** again so that the **.bashrc** file executes to set the new **PATH**.

6.11. Now, type the **echo** command again:

```
$ echo $PATH
```

6.12. Write, in your notebook, the difference(s) in the output you obtain now compared to earlier (see step 6.4.)

7. Downloading your chosen Dining Philosophers Problem code and configuring the Makefile

7.1. Using the command-line, change into your **programs** directory, and create a directory for your Dining Philosophers Problem code; for e.g.: you could call it **diningphil**.

7.2. Access the page on the ThreadMentor e-book website:

https://pages.mtu.edu/~shene/NSF-3/e-Book/index.html

7.3. Access the page corresponding to the solution you have chosen. You need the page with the code listings.

7.4. For each code listing, you will notice captions that state "**Click here to download this file**". **DO NOT Left-click these**. Instead, Right-click on the "**here**" in each caption and select "**Save Link as…**". Save each file to the Dining Philosophers subdirectory you created in your **`programs`** directory in step 7.1. above.

7.5. Each solution has the following *three* files. Check that you have them. If you don't, let your Lecturer know:

- Two C++ files (files ending in .cpp)
- One header file (file ending in .h)

7.6. Access the OS BrightSpace page again. Under the **Project** unit, you will see a number of sub-units starting with "**Dining Philosopher: Makefile for…**" and then a named solution. Click on the **Makefile** in the sub-unit relating to your chosen solution. Click **Download** and save the file, changing its name simply to **Makefile**; i.e. remove the name of the solution from the file name. **This is very important!**. Also, make sure you save it in the *same* directory as the code files of your chosen solution.

7.7. For those students using my Mageia 8 i586 32-bit appliance VM, with the **student** account, there is no need to do this step. If you set up Mageia 8 i586 32-bit yourself with your own user account, then you need to edit the Makefile and replace the username **student** with *your* username.

8. Compiling and Running your Dining Philosophers Solution

8.1. To compile the code for your solution, simply type:

```
$ make
```

8.2. The compiler will output quite a number of **warnings**, but there should not be any **errors**. List the contents of the directory. If the compilation has been successful, you should see two "`.o`" files (object files) and one green file with a "*" appended to the end of the filename, to indicate that it is executable.

8.3. The program for each solution requires *specific* arguments to run. Using the method shown in Practical 02, run the program on the command-line, so that it will output a message telling you what the required arguments are. If you need assistance with this, ask your Lecturer.

8.4. Now, re-run your program with the *correct* arguments. The main ThreadMentor window should display with an informational window over it – click **OK** on it. Check that you can open the **History Graph** window, **Thread Status** window and then click on the **Start** button. Some output should also appear in the console.

8.5. If you see an error stating *"**Failed to load visual system**"*, this means that you have not completed Section 6. correctly. Please work through that section again, following the steps carefully. If you see other errors, please ask your Lecturer for help!

8.6. You are now set up to run your Dining Philosophers Problem solution. Well done!

9. Next Steps in your Project

9.1. You and your team now need to do some more reading. Some questions that would be useful for you to address are:

1. What are all those different ThreadMentor tags?
2. What other windows and buttons are available in ThreadMentor, what do they do and what information do they provide?
3. What's a Mutual Exclusion Lock?
4. What's a Semaphore?
5. How do they differ?
6. What does my solution use?
7. What problems, if any, does our solution contain or solve?