Lab 0: Mac Addendum

What You need if you just bought a Mac

Many things are harder at the beginning, until you learn them. On the other hand, you will be better prepared for using Linux in the future or becoming a power Windows user. Windows operating systems users can delay learning these things, but will eventually need to get there, too.

This document assumes you are starting from zero. If there is more you need, let us know and we can add more information.

First, know what you are using (always good advice). Advise on the internet is often outdated and only helps once you have a lot of knowledge.

Find out and write dows (In the Notes application) The following

- Select > About This Mac Note the operating system (e.g. macOS Monterey Version 12.5.1), the Processor (e.g. 2.9 GHz 6-Core Intel Core I9), Memory (e.g. 32 GB 2400 Mhz DDR4)
    - Most critical is whether the processor is Intel or Arm (which may be M1 or M2)
    - Intel is also described other places as x64, x86-64, AMD64, Intel 64, or nothing (just assumed)
    - Arm is also described other places as AArch64, arm64, M1 (or other specific chips)
- What shell are you using
    - Open a terminal window (launchpad > Terminal) and read the shell from the menu bar - default is zsh (which is assumed here) but bash is another possibility for older Macs

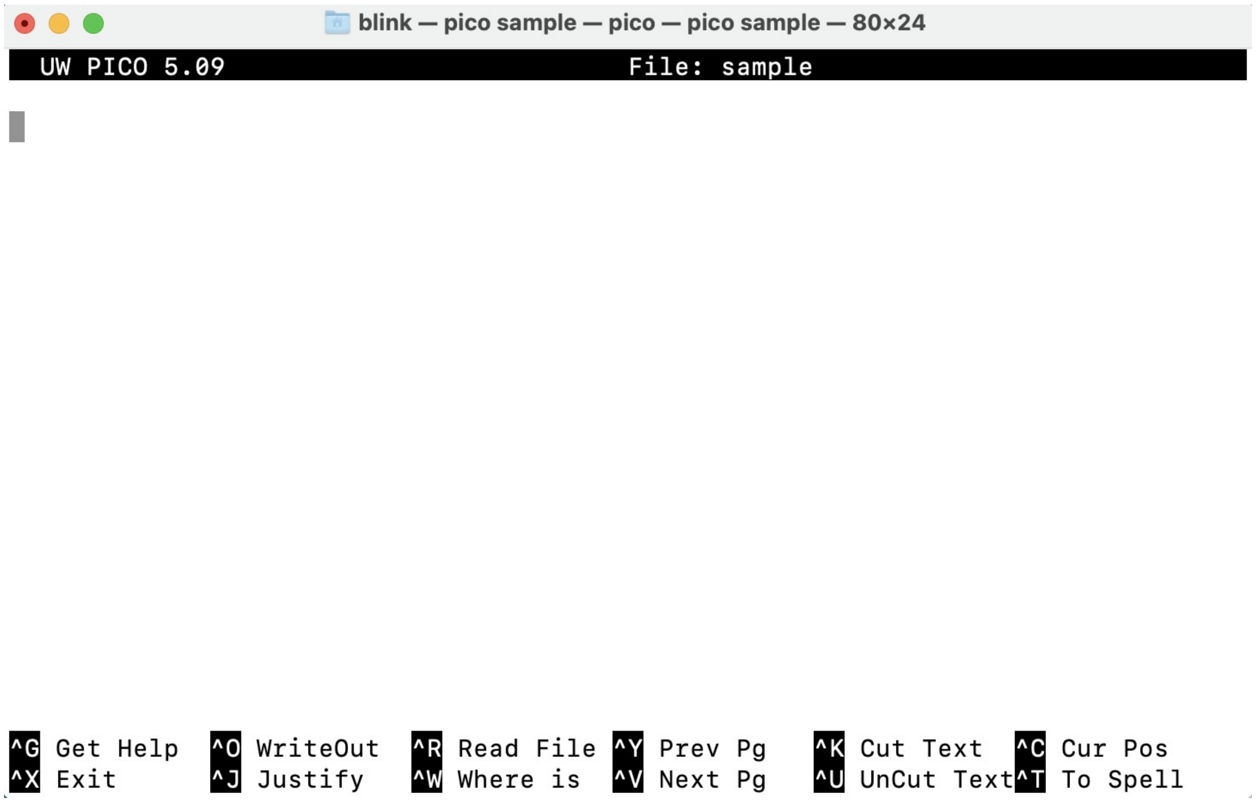
All the software installed will need to work on the processor chip you have, so you need to choose appropriate downloads. This includes the Java JDK. Some, such as ant are independent of architecture.

You will need to familiarize yourself with the terminal window. A reasonable tutorial is available from Apple: Terminal User Guide. Learn the material in the “Work in Terminal windows” section

You will also need a text editor. Macs come with some basic ones, you probably want to graduate to something better (see lots of opinated articles, such as Best Text Editors 2022. In reality it is a matter of taste and you can start small with nano or pico (pico seems to be the default since MacOS 12.3 onward and is what you get if you try to use nano as well, unless you install nano yourself). Opening .txt files in finder is a separate issue than using the command line. If interested, see How to Change Default Text Editor on the Mac.

Learn pico:

- Open a terminal window
- Type `pico sample`
- The name of the file will be “sample”. Files do not have to have extensions, which are just part of a file name preceded by a period. In the file name “sample.txt” the part “txt” is often called an extension. This terminology comes from MS Windows via MS DOS but is a convenient signal of what a file might contain. It is also used as a sign to a windowing system as to what applications to use to open the file. It does not guarantee the actual file contents, however.
- You should see the terminal window changing to look similar to this:

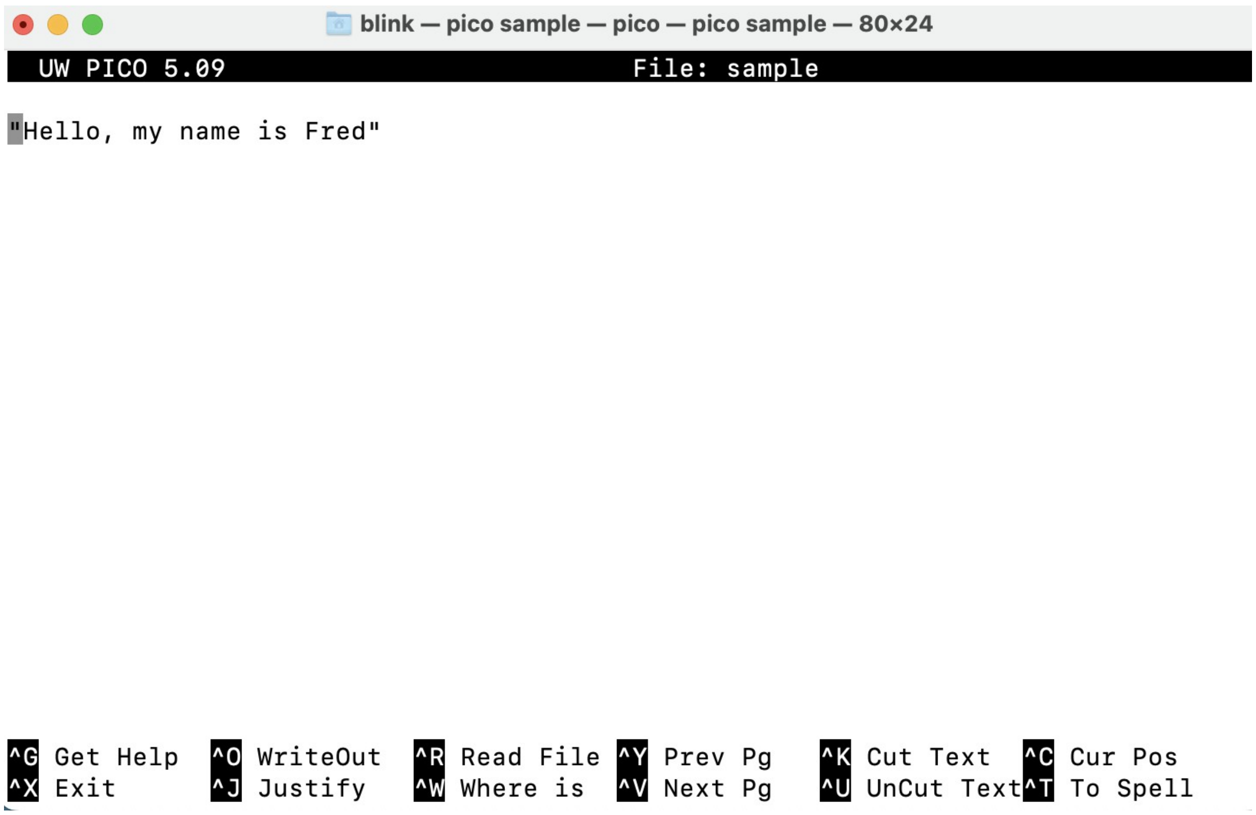


```
blink — pico sample — pico — pico sample — 80×24
  UW PICO 5.09                          File: sample




^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Pg   ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is  ^V Next Pg   ^U UnCut Text^T To Spell
```

- Notice the name of the editor and file shows at the top, and a cursor is shown where you can type. Nothing will be saved to the file until you enter control-o (shown ^O at the bottom) followed by return. Note that even though the control charters are shown with capital letters you do not use the shift key. To type a control character hold down the control key and simultaneously press the other key.
- Type “Hello, my name is Fred” and save the file. Then exit (^X) back to the command line.
- Verify the contents of the file using the cat command: `cat sample` You should see exactly what you typed.
- Now open the file again with the command `pico sample` Since you just typed this you can use the up arrow key to back up to the earlier command rather than typing it again.



```
  UW PICO 5.09                                  File: sample

"Hello, my name is Fred"




^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Pg    ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where is   ^V Next Pg    ^U UnCut Text ^T To Spell
```

- Whatever you typed should be shown and the cursor (showing where the next character you type will go) should be at the top left, as shown.
- You can move around in the text you have typed using the arrow keys, but not past what you have typed. The delete key will delete the character to the left of the cursor.
- This is all you must know to use pico.
- The most common commands you can execute in pico are listed at the bottom. Type ^g and read through the help.

Basic commands and defining environment variables

Environment variables are names that can hold values which are available across your system. Any program can access them. Often, they need to be defined as part of a software installation, sometimes manually so it is necessary to know how to do this.

Unfortunately, the details will depend on the shell you are using to run commands in the terminal window. When you start a terminal window, it reads the environment variables from files which depend on the shell. This document will describe the process for zsh. (again, be careful, many internet articles will assume you are using bash and have guidance that will not work – often a top hit will be from 2015!)

So open a terminal window (have it on the tool bar, see Customize Toolbar if it is not there)

- Learn basic commands `man`, `pwd`, `ls`, `cd`, `mkdir`, `rm`, `cat`, `echo` and understand the file system.
    - File system is rooted at a folder (folder or directory are synonyms. Each OS has a slight preference to one of these) called /

    - The character / is also used as a folder separator in a path that takes you to a file. Folders are special types of file (which can hold lists of files [including folders])
    - Your login folder is /User/*name* where *name* is your login name. You can refer to it as ~ or $HOME. In this example, User and *name* are folders.
    - The command `man` ***`command`*** describes how the *command* works, e.g. `man ls`
    - Command can have arguments (separated by white space). An argument may be an option (starting with - or --) or a file name
    - The command `pwd` tells you what your current folder is.
    - The command `ls` lists the files in your current folder
        - `ls –l` also list file information (the first character is d for a folder, - for a regular file)
        - `ls –a` lists hidden files as well (files whose name starts with a period)
        - `ls –al` does both
    - `cd` *`folder`* changes the current folder to its argument.
    - `mkdir` *`folder`* creates a folder named in the command
    - `rm` removes a file.
    - `cat filename` writes the contents of a text file to the terminal (cat can also concatenate files).
    - `echo expression` evaluates the expression and writes the results to the terminal
    - And there are lots more commands
- Be sure you are in your home folder with the following command:
    - `cd ~`
- Open/create the zsh configuration file:
    - `pico .zshrc`
- Define a sample environment variable by entering the following line in `.zshrc`:
    - `export TEST_ENV=”it worked”`
    - No spaces around the = sign!
- Save the file and exit pico.
- Now, given an environmental variable, putting a $ before it evaluates it and returns the result.
    - Try the command `echo $TEST_ENV`
    - Nothing is written.
    - This is because environment variables in `.zshrc` are only read when you open a terminal window.
    - Close the terminal window (^D at the beginning of a line will do this)
    - Open a new window and try again.
    - `it worked` should be written to the terminal window.
- Lab 0 will want you to define JAVA_HOME and ANT_HOME appropriately.

Commands need to be found and operating systems have a search path of folders to look in to find commands, generally in and environment variable called PATH.

- Each folder in the search path is separated from the others by the : character (; on Windows)
- See your current PATH (`echo $PATH`)
- You can add something to the path by redefining the new path in terms of the current path
- Examples are in Lab 0. Again edit .zshrc and add an export to modify the path
    - `export PATH=$PATH:`*`newFolder`* to put it at the end of the current path
    - `export PATH=`*`newFolder`*`:$PATH` to put it at the beginning of the current path

This should be a start. There is always more to learn.