Tools

TABLE OF CONTENTS

Before continuing, we will take a brief break to introduce you to some useful tools that make a good fit in any system hacker's toolbox. Some of these (git, make) are MANDATORY to understand in that you won't be able to compile/submit your code without understanding how to use them. Others such as gdb or tmux are productivity boosters; one helps you find bugs and the other helps you multitask more effectively. All of these come pre-installed on the provided virtual machine. They are ESSENTIAL.

Note: We do not go into much depth on how to use any of these tools in this document. Instead, we provide you links to resources where you can read about them. We highly encourage this reading even though not all of it is necessary for this assignment. We guarantee you that each of these will come in handy throughout the semester. If you need any additional help, feel free to ask any of the TA's at office hours!

git

Git is a version control program that helps keep track of your code. GitHub is only one of the many services that provide a place to host your code. You can use git on your own computer, without GitHub, but pushing your code to GitHub lets you easily share it and collaborate with others.

At this point, you have already used the basic features of git, when you set up your repos. But an understanding of the inner workings of git will help you in this course, especially when collaborating with your teammates on group projects.

If you have never used git or want a fresh start, we recommend you start here. If you sort of understand git, this presentation we made and this website will be useful in understanding the inner workings a bit more.

make

make is a utility that automatically builds executable programs and libraries from source code by reading files called Makefiles, which specify how to derive the target program. How it does this is pretty cool: you list dependencies in your Makefile and make simply traverses the dependency graph to build everything. Unfortunately, make has very awkward syntax that is, at times, very confusing if you are not properly equipped to understand what is actually going on.

A few good tutorials are here and here. And of course the official GNU documentation (though it may be a bit dense) here .

man

man — the user manual pages – is really important. There are lots of stuff on the web, but the documentation in man is definitive. The man pages can be accessed through a terminal. For instance, if you wanted to learn more about the ls command, simply type `man ls` into your terminal. If you were curious about a function called `fork`, you could learn more about it by typing `man fork` into your terminal.

gdb

Debugging C programs is hard. Crashes don't give you nice exception messages or stack traces by default. Fortunately, there's the GNU Debugger, or gdb for short. If you compile your programs with a special flag `-g` then the output executable will have debug symbols, which allow gdb to do its magic. If you run your C program inside gdb, you will be able to not only look at a stack trace, but also inspect variables, change variables, pause code and much more! Moreover, gdb can even start new processes and attach to existing processes (which will be useful when debugging PintOS.)

Normal gdb has a very plain interface. So, we have installed cgdb for you to use on the virtual machine, which has syntax highlighting and few other nice features. In cgdb, you can use `i` and `ESC` to switch between the upper and lower panes.

This is an excellent read on understanding how to use gdb. The official documentation is also good, but a bit verbose.

tmux

tmux is a terminal multiplexer. It basically simulates having multiple terminal tabs, but displays them in one terminal session. It saves having to have multiple tabs of sshing into your virtual machine.

You can start a new session with `tmux new -s <session_name>`.

Once you create a new session, you will just see a regular terminal. Pressing `ctrl-b + c` will create a new window. `ctrl-b + n` will jump to the nth window.

`ctrl-b + d` will "detach" you from your tmux session. Your session is still running, and so are any programs that you were running inside it. You can resume your session using `tmux attach -t <session_name>`. The best part is this works even if you quit your original ssh session, and connect using a new one.

Here is a good tmux tutorial to help you get started.

If you use macOS and iTerm2, you can utilize iTerm2's built-in tmux integration, which — when you are SSH'ed into your VM — will automatically perform the appropriate tmux commands for you whenever you create a new tab or split using iTerm2's normal tab/split commands (`Cmd+t` and `Cmd+d`). To use this, first enable iTerm2's tmux integration by following these instructions, then after SSHing into your VM, add the following lines to the end of your `~/.bashrc`.

```
if [ -n "$SSH_TTY" ] && [ -z "$TMUX" ] && shopt -q login_shell
then
tmux -CC new -A -s main
fi
```

The next time you SSH into your VM, the tmux integration should be working.

vim

vim is a nice text editor to use in the terminal. It's well worth learning. The vimtutor program provides an excellent, beginner-friendly tutorial on how to use vim. You can open it by simply running vimtutor inside your VM (note when you first run it, it may display a message along the lines of "The file /usr/share/vim/vim80/tutor/tutor.C... does not exist, creating english version..." – this is okay, just be patient for a few seconds and the tutorial will open). Others may prefer emacs. Whichever editor you choose, you will need to get proficient with an editor that is well suited for writing code.

If you want to use Sublime Text, Atom, CLion, or another GUI text editor, look at Editing code in your VM, which shows you how to access your VM's filesystem from your host.

ctags

ctags is a tool that makes it easy for you to navigate large code bases. Since you will be reading a lot of code, using this tool will save you a lot of time. Among other things, this tool will allow you to jump to any symbol declaration. This feature together with your text editor's go-back-to-last-location feature is very powerful.

Instructions for installing ctags can be found for vim here and for Sublime here. If you don't use vim or Sublime, ctags still is probably supported on your text editor although you might need to search installation instructions yourself.
