CMPS 111 - Spring 2017: Assignment 0

Assigned: April 04 at 17:00

Due: Tuesday, April 11 at 15:00

Unlike later assignments, this assignment is pass / no pass. **You must pass it to pass the course.** It won't otherwise impact your final grade, though you'll have a lot of difficulty with later assignments if you have a lot of difficulty with this one.

Goals

- Install set up the FreeBSD environment in a Virtual Machine (VM) correctly.
- Set up your Git credentials and repository on the server and VM.
- Write a simple C program mycat, that has the same behavior as the built-in Unix cat program.

Files for this assignment, as for every other assignment in the class, will be submitted by committing them to your local git repository and pushing the changes to the git server.

Installing a FreeBSD VM

Please use the "FreeBSD-10.3-RELEASE-amd64-disc1.iso" from here, to ensure that you're using the exact same version as everyone else in the class.
Use the "FreeBSD-10.3-RELEASE-i386-disc1.iso" from here only if you do not have an x86_64 CPU and an x86_64 host operating system (If you have an ancient, >8 years old computer).

The following instructions are for installing a FreeBSD VirtualBox VM. If you are more comfortable with some other hypervisor like VMware Workstation/Fusion, feel free to use that.

Creating and Configuring the VM in VirtualBox

1. Create a new Machine in VirtualBox, from Machine -> New, or click the New button.
2. Select the Type as BSD and Version as FreeBSD (64-bit). If you do not have the option for 64-bit then, you can select the 32-bit one but use the i386 image mentioned above.
3. Assign 2048 MB memory for the VM, assuming your computer has 4 GB RAM on the system. You can assign more if you wish. It can speed up your kernel build time. If do not have 4 GB of memory on your computer, you can get away with 1024 MB (or even lesser) of memory for the VM; but the kernel will take longer to build.
4. Create a VDI disk file of at least 10 GB, but it is recommended you create a 16 GB one if you have the storage space on your computer. Select the fixed size allocation option, if you can.
5. Select the VM from the list and open its Machine Settings, from Machine -> Settings, or click on the Settings button.
6. Go to Storage, and attach the downloaded .ISO file to the virtual optical drive there. The optical drive is the one with the disc icon, and you can attach an ISO using the disc button there to select a virtual optical disc file.
7. Go to Network, the first adapter must be attached to NAT.

8. In the Network tab under Advanced, select Port Forwarding. Enter the following information as a rule and click on OK:
    - Name: SSH
    - Protocol: TCP
    - Host IP: *(leave empty)*
    - Host Port: 2022 (Or any number above 2000 and less than 65536). Pick one that you will remember.
    - Guest IP: 10.0.2.15
    - Guest Port: 22
9. Save the Machine Settings by clicking on OK. You can now start the installation process.

Booting from the ISO file and installing FreeBSD

You're now booting from the ISO file that you downloaded. You'll go through the FreeBSD install script, making selections to configure your operating system. In general, <ENTER> is used to confirm a selection (moving to the next screen), the arrow keys and space bar are used to navigate to an item and select / unselect it and the <TAB> key to switch between UI contexts.

1. Hit <ENTER> to boot the default kernel on the installation disk.
2. Hit <ENTER> to use the default keyboard layout (US).
3. Choose a hostname. This can be anything you want, but should start with a letter and contain only letters, numbers, dashes, and underscores. Hostnames are case-insensitive.
4. Select the distributions you want: only install lib32 and ports (the ones with next to them are selected). *Do not install system source code*!
5. Next, ensure that you're using Auto (ZFS) for partitioning.
6. For ZFS configuration, swap size should be 512MB, but the other defaults are fine.
7. For the Pool Type / Disks option, select "Stripe – No Redundancy".
8. When asked which disk to use, there should only be a single choice (the virtual disk). Use it.
9. Yes, you want to destroy the contents of your virtual disk. After you hit <ENTER>, the system will copy data from the (virtual) CD to your (virtual) disk. This may take a few minutes—be patient!
10. Select a root password when asked. Write it down! There's no sysadmin besides you who can help if you forget it.
11. For network configuration, you do want IPv4 and DHCP, but you don't need IPv6 as that does not work on campus. It might work on other networks, but it is unnecessary for the purposes of this class.
12. Your search domain should be empty (erase whatever is present there).
13. For time zone, select NO because you don't know if you're using UTC. Then, select the proper time zone (America -> United States -> Pacific Time).
14. For system configuration, enable sshd, dumpdev, and ntpd.
15. Yes, you'd like to add a user. For username, use your CruzID because it'll simplify things. When it asks you if you wish to add user to additional login groups, enter wheel. For shell, pick any of sh, csh, or tcsh. Pick defaults for everything else, and pick a good password. Again, write it down!
16. Then reboot the VM (not your computer).
17. While it is rebooting, eject the disk image (.ISO) file by selecting Devices -> Optical Drives -> Remove disk from virtual drive on the VM window.

Required packages

pkg install git

This installs the git system, which you'll obviously need.

pkg install virtualbox-ose-additions

This installs the VirtualBox guest additions for FreeBSD. You'll have to add the following lines to /etc/rc.conf.

vboxguest_enable="YES"
vboxservice_enable="YES"

Optional packages

If you have no opinion on the following optional packages, install sudo and vim.

$ pkg install sudo

This makes it easier to run things as root. Use visudo to edit the sudo file, and uncomment the line immediately following this one:
Same thing without a password
by removing the # from the front of the line.

pkg install emacs-nox11
pkg install vim

Which is good if you like emacs **/** vim. vi is already installed for you. Install vim if you are more comfortable with that.

pkg install bash

Which is good if you like bash. tcsh is already installed for you, and is the default shell. You're welcome to install other packages if you like. You can change the default shell for your user using chsh.

Setting up your repository and git

Next, set up your repository. You'll write all your code here, and use git to track changes and submit them for grading. Please read this page for information on how to set up your git account. Once that's done, clone your repository from the CMPS111 git server to your FreeBSD VM, preferably logged in as the user you created above (**Never use git as root**).

If you are not well versed with the Unix terminal or the use of Git, you will need to be to successfully complete the assignments. Read the git resources mentioned above to become familiar with git. You'll be using it a lot. The goal of the assignments is to also familiarize

yourself with the tools used for real-world kernel programming. Feel free to scour the internet to learn more about the Unix terminal, how to use the command line programs like cp, mv, vim, less etc. SSH into the VM from your host computer using the port forwarding details entered above:

$ ssh user@localhost -p 2022

You will need to use git branching for every assignment, including this one. So, it is strongly recommended that you also familiarize yourself with git. You can start with the overview here, but it is also recommended that you understand what git is and how to use it from here. You'll need to understand that for future assignments, especially for the group ones.

Get in the habit of committing to git frequently. Your commits will be stored locally until you push them to the server, which you can do any time you want. Note that, if you commit your changes but don't push them to the server, your grader can't see them. While we're going to use your commit times to determine if your assignment is on time, you need to push your changes to the server within an hour of the actual due date. So, commit often and push often.

The cat program

The only code you must write for this assignment is to implement the basic cat program, without support for any flags. That means your code needs to copy data from each of the files specified on the command line to standard output. For example,

$ ./mycat file1 file2 file3

will copy all the data from file1, file2, and file3 to standard output, in that order. Note that the data might be binary; your code should work in that case. Process files one at a time; if mycat can't open a file to read it, the program should stop and print an error message to standard error. This may mean that some of the files were copied, and others not—this is fine, and is what the real cat program does.

Before writing code for this assignment, as with every other assignment, you should write up a design document. Your design document should be called design.txt (if plain text), or design.pdf (if PDF) and should reside with the rest of your code in the asgn0 directory on your asgn0 git branch in your git repository. Formats other than plain text or PDF are not acceptable; please convert other formats (Word, LaTeX, HTML, ...) to PDF. **Don't submit a Word file—it's not a plain text file.** Your design document should describe the design of your code in sufficient detail using plain English and some pseudocode, using which an experienced C programmer can duplicate your work. This includes descriptions of the data structures you use, all non-trivial algorithms and formulas, and a description of each function including its purpose, inputs, outputs, and assumptions it makes about the inputs or outputs. The format of the design document is at your discretion. While the design of the mycat will be quite short and even trivial, this is for you to obtain experience writing one. Write up a design document **before** you start writing code. You can update the design document as you write code if necessary.

The cat program is very short—fewer than 50 lines of code, most of which are error handling. You must to use the open(), close(), read(), and write() system calls (see the man pages for

them) as well as perror() to write to STDERR and print out an error message. Usage of these system calls is like using functions in C. You can also see the man pages for these system calls on the internet. For example, here is the one for open(). Anything other than these system calls would mean your cat program will not work for all the below use cases.

You will need to support the following use cases for your cat program:
$ ./mycat file1 file2 file3
$ ./mycat
$ ./mycat < file1 > file2
$ ./mycat file1 | grep string1

Your program must be written in C (not C++, not Java), and must be compiled by a Makefile that creates an executable called mycat. Use the default C compiler for FreeBSD (That is not gcc, by the way). That is preinstalled on a FreeBSD system.

Your mycat program must compile and run as-is on an unmodified FreeBSD installation. So, mycat executable must be created by simply typing in
$ make
in the asgn0 directory on your asgn0 branch in your git repository.

Deliverables:

- mycat.c, design.pdf (or design.txt), Makefile and Readme.txt in the asgn0 directory on the asgn0 branch in your git repository.
- Submit the Git commit ID on eCommons. It is a universally unique SHA-1 checksum that represents a git commit. You can see the latest one on the current branch by typing in:
  $ git log -n 1

Grading:

This assignment is a Pass/Fail individual assignment. There will be no points earned by this assignment contributing to your overall grade. If you fail this assignment you will be asked to drop the course.