

Lab #1
Installing a System
Due Friday, September 6, 2002

Name: _____

Lab Time: _____

Grade: _____/10

The Steps of Installing a System

Today you will install a software package. Implementing a software system is only part of a software engineer's job. Once implemented, a system needs to be installed. A software engineer needs to design the installation process, also called deployment, to make it as easy as possible for the system's users to obtain and install the system.

As you install this lab's example program, think about what you like and dislike about the process. What aspects can be improved and how? The steps to install a Unix program typically involve:

- a. Finding
- b. Getting
- c. Unpacking
- d. Localizing
- e. Building
- f. Installing
- g. Sanity Testing

Finding and Getting

For this lab we will be installing a program called gnuchess. It is a chess program created by the GNU software project. A copy of the distribution for gnuchess can be found at the location below. Copy this file into your tmp directory.

```
~/csci3308/src/chess-5.02.tar.gz
```

Unpacking

The gnu chess distribution was “packed” with a program called “tar”, and then compressed with a program called “gzip.” In order to “unpack” the distribution, you will have to undo these steps in the reverse order. You will first decompress the distribution with gzip and then unpack it with tar. However, first we will review the tar command.

The tar command takes the form

```
tar c|r|u|x|t[option...] [file...]
```

This line is in the standard format for man pages. The vertical bar means exclusive or. That is, the word `tar` must be followed by one *and only one* of `c`, `r`, `u`, `x`, or `t`. The square brackets mean optional, and the ellipsis means repetition. So the first letter can be optionally followed by zero or more options and zero or more files. Also notice that there is no space between the first letter and the options, but there is a space between the options and the files. The following are examples of tar commands.

```
tar x
tar cv file1 file2
```

So now you know the syntax of tar commands, but what do they do? What tar does depends on the first letter.

- `c` Create. Tar creates a tar file containing the files listed on the command line. If a directory is listed all of its files and subdirectories are included as well.
- `r` Replace. Similar to create, it adds files to an existing tar file
- `u` Update. Similar to replace, except a file is only stored in the archive if it has a timestamp that is more recent than the existing copy stored in the archive.
- `x` Extract. Extracts from the tar file the files listed on the command line. If no files are listed it extracts the entire tar file.
- `t` Table of contents. Similar to extract except that tar only lists the names of the files that would be extracted if `x` were used, but performs no extraction.

Another important question is which tar file does tar use? The tar file is specified by the option `f` followed by the name of the tar file. Notice that this is an optional argument. `tar` stands for `t`ape `a`rchive. It was originally intended to be used with a tape drive. If no `f` option is given tar assumes you want to read or write a tape drive. This usually results in tar reporting that it can't find the device file for the tape drive, or on some versions tar will sit quietly and wait for the drive to become available, and you will wait thinking tar is doing something when it is not. Because the `f` option is so important I will rewrite the tar description this way.

```
tar c|r|u|x|t[option...][f tarfile] [file...]
```

The only other option that I will tell you about is the `v` option that stands for verbose. With verbose, tar prints more information that it would otherwise. Most other options deal with the characteristics of tape drives.

1. Change to your `tmp` directory where you copied the distribution.
2. To uncompress the gzipped file type

```
gunzip chess-5.02.tar.gz
```

The command `gzip -d chess-5.02.tar.gz` will also work. The `-d` flag stands for decompress. There is more than one way to skin a cat...especially in Unix! The distribution file is now just a tar file: `chess-5.02.tar`.

3. Before you unpack the tar file you want to know what is in it. Type:

```
tar tvf chess-5.02.tar
```

4. Describe the parts of this tar command.

5. This tar file is a source distribution. That is, it contains only source code. As such, it needs to be unpacked into your source directory (`src`.) When you unpack this distribution you want it to go in its own directory under your source directory. Fortunately, the tar file was packaged so that it would create a new directory and place all of the files in that directory. Go into your source directory and unpack the distribution with the command:

```
tar xvf ~/csci3308/tmp/chess-5.02.tar
```

6. Now you have all of the source code in your source directory so you can delete the tar file in your tmp directory. The tar file is only used for a short time which is why you put the file in the tmp directory in the first place. Every single person in this class will be downloading the same tar file, and it is a rather large file so delete it now.

```
rm ~/csci3308/tmp/chess-5.02.tar
```

Localizing

7. You now have a new directory under your `src` directory containing the source code for the `gnuchess` program. Enter this directory and take a look around. There are several subdirectories, including one named `src`. Enter the `gnuchess` source directory and look at its contents. Also take a look in the `doc` directory. A distribution generally includes a few files with names in all capital letters (such as `README` or `CHANGES`) that you should read before proceeding with the installation. Look over these files now.
8. Which file gives instructions on how to install the program?
9. You will be leaving the source code in this directory, and building from a different directory as described in the instructions. Change to the directory `~/csci3308/arch/$ARCH/build`. Notice that you are building in your architecture specific build directory. `$ARCH` automatically selects the architecture of the machine you are on. You will be able to repeat this process for other architectures in the lab and have versions of `gnuchess` that will work no matter what computer you are on.
10. Now, create a subdirectory in your `build` directory for `gnuchess`. Call this directory `gnuchess`, and change to the new directory. Type the following all on one line:

```
~/csci3308/src/chess-5.02/src/configure  
--prefix=$HOME/csci3308  
--exec-prefix=$HOME/csci3308/arch/$ARCH
```

You should now have several files in your directory including one called **Makefile**.

Building

In this same directory type **make**. This will take some time, and there will be a lot of output generated to your screen. Some of the output may be warnings. You can ignore these warnings. Gnuchess should still build correctly. All of the intermediate files, and the gnuchess executable should now be in this directory.

Installing

Before you install type the following:

```
which gnuchess
man gnuchess
```

11. These commands will report that they cannot find the gnuchess program or its man page, despite the fact that you have just finished building gnuchess from its source code.
12. Why can't the shell find these things?

13. The “**which gnuchess**” may report that it has found another installation of gnuchess. If so, explain why it found that version and not the one that you compiled.

14. The directory `~/csci3308/man/man6` is the correct location to install the Gnuchess man page, but that directory doesn't exist yet, and the gnuchess makefile doesn't create it correctly. Go into your directory `~/csci3308/man`, and create a directory called **man6**. Man pages in this directory will be in section 6 of the manual.

15. Now return to the build directory where you just built gnuchess. type `make install`. Some of the files that you created when you did the building step will be copied to their final destinations so that you can run gnuchess. Now type the following:

```
rehash
which gnuchess
man gnuchess
```

If you added the directories to your path and MANPATH correctly in lab 0, you should now see the gnuchess program and documentation.

Sanity Testing

After you install gnuchess you should make sure that it functions properly. This process is called sanity testing. The purpose of sanity testing is not to find bugs in a program. Rather, it is to check that a program has been installed correctly. Generally, sanity testing involves using the program at least once to make sure that it appears to operate as expected. However, sanity testing can be a much more involved process in which a test suite is run to confirm that a new version of a program can still provide the same functionality that a previous version did.

Type `gnuchess`. gnuchess will print a prompt waiting for your first command. Now type `go` and the computer will calculate the first move for the white player. Once it is done, it will print a chess board displaying its results. Type `go` again to have the computer calculate a move for the black player. You can continue typing `go` if you want, but for our purposes the sanity test is now complete. To exit gnuchess, simply type `exit`.