What is a port

The Ports Collection is essentially a set of Makefiles, patches, and description files placed in /usr/ports.
The port includes instructions on how to build source code, but does not include the actual source code. You can get the source code from a CD-ROM or from the Internet. Source code is distributed in whatever manner the software author desires. Frequently this is a tarred and gzipped file, but it might be compressed with some other tool or even uncompressed. The program source code, whatever form it comes in, is called a ``distfile''
A port skeleton is a minimal set of files that tell your FreeBSD system how to cleanly compile and install a program.

Composition of a sample port.

A “Makefile”.
The Makefile contains various statements that specify how the application should be compiled and where it should be installed on your system.
A “distinfo” file.
This file contains information about the files that must be downloaded to build the port and their checksums, to verify that files have not been corrupted during the download using md5.
A “files” directory.
This directory contains patches to make the program compile and install on your FreeBSD system. Patches are basically small files that specify changes to particular files. They are in plain text format, and basically say ``Remove line 10'' or ``Change line 26 to this ...''. See the man pages for diff and patch for details.
A “pkg-descr” file.
This is a more detailed, often multiple-line, description of the program.
A “pkg-plist” file.
This is a list of all the files that will be installed by the port. It also tells the ports system what files to remove upon

Installing using Ports

In our example we are going to install the lsof package which is a nice systems administration utility that does not come with FreeBSD by default.
However before we do this we need to make a small change to our /etc/make.conf.

- vi /etc/make.conf
- We are going to insert a magic line in here which you won't need at home:

```
MASTER_SITE_OVERRIDE=ftp://inst.ws.linuxchix.or.ke/pub/FreeBSD/ports/distfiles/
```

Magic explained..

If you have a local mirror of the FreeBSD ports, you can save bandwidth on your network by configuring you machines to try downloading from that mirror first.
You can do this on a per port basis.. e.g

```
# cd /usr/ports/sysutils/lsof
# make MASTER_SITE_OVERRIDE=.... install
```

However, putting this setting in /etc/make.conf ensures we won't need to type this long command all week.

Installing the lsof port

Now that we have edited our /etc/make.conf, let us install the lsof port:

```
# cd /usr/ports/sysutils/lsof
# make
# make install
```

Once you are returned to your prompt, you should be able to run the application you just installed. Since lsof is a program that runs with increased privileges, a security warning is shown. During the building and installation of ports, you should take heed of any other warnings that may appear.

You can save an extra step by just running make install instead of make and make install as two separate steps.

Note: If you are using the c shell (csh) or it's relatives (tcsh) you will have to ask it to regenerate the internal database of binaries and where in your $PATH they are located. So, if after installing a new port you can't run the binary, then do “rehash” to fix that.

For users which cannot be connected all the time, the make fetch option is provided. Note that if a port depends on libraries or other ports this will not fetch the distfiles of those ports too. Replace fetch with fetch-recursive if you want to fetch all the dependencies of a port too.

Note: You can build all the ports in a category or as a whole by running make in the top level directory, just like the aforementioned make fetch method. This is dangerous, however, as some ports cannot co-exist. In other cases, some ports can install two different files with the same filename.

Using the Ports Collection will use up disk space over time. Because of this tendency of the ports tree to grow in size, after building and installing software from the ports, you should always remember to clean up the temporary work directories using the make clean command. This will remove the work directory after a port has been built and installed. You can also remove the source distribution files from the distfiles directory, and remove the installed ports if the need for them has passed.

Removing Installed Ports

```
# cd /usr/ports/sysutils/lsof
# make deinstall
```

That was easy enough. You have removed lsof from your system. If you would like to reinstall it, you can do so by running make reinstall from the /usr/ports/sysutils/lsof directory.

Note: The make deinstall and make reinstall sequence does not work once you have run make clean. If you want to deinstall a port after cleaning, use pkg_delete

Updating the ports tree

This is a quick method for getting the Ports Collection using CVSup. On Monday we installed the cvsup-without-gui from the CDs. If you don't have it installed, then install net/cvsup

As root, edit /usr/share/examples/cvsup/ports-supfile
Change CHANGE_THIS.FreeBSD.org to a CVSup server near you. In our case this will be noc.ws.linuxchix.or.ke.

Run cvsup:

```
# cvsup -g -L 2 /usr/share/examples/cvsup/ports-supfile
```

It is wise to do this regularly in order to ensure that you install the latest version of a port each time you use ports for installation. The same command is used to update your /usr/src tree; just that you use the stable-supfile.

UPGRADING PORTS

Keeping your ports up to date can be a tedious job. For instance, to upgrade a port you would go to the ports directory, build the port, deinstall the old port, install the new port, and then clean up after the build. The sysutils/portupgrade utility will do everything for you! Just install it like you would any other port, using the make install clean command.

```
# cd /usr/ports/sysutils/portupgrade
# make install
```

Now create a database with the pkgdb -F command. This will read the list of installed ports and create a database file in the /var/db/pkg directory.

```
#pkgdb -F
```

Now when you run portupgrade -a, it will read this and the ports INDEX file. Finally, portupgrade will begin to download, build, backup, install, and clean the ports which have been updated. portupgrade comes with a lot of options for different use cases, the most important ones will be presented here:

Portupgrade Options

If you want to upgrade only a certain application, not the complete database, use portupgrade pkgname, include the flags -r if portupgrade should act on all those packages depending on the given package as well, and -R to act on all packages required by the given packages.

To use packages instead of ports for installation, provide -P. With this option portupgrade searches the local directories listed in PKG_PATH, or fetches packages from remote site if it is not found locally. If packages can not be found locally or fetched remotely, portupgrade will use ports. To avoid using ports, specify -PP.
To just fetch distfiles (or packages, if -P is specified) without building or installing anything, use -F.

Note: It is important to regularly update the package database using pkgdb -F to fix inconsistencies, especially when portupgrade asks you to. Do not abort portupgrade while it is updating the package database, this will leave you an inconsistent database.

Portsnap: The future.

- As of FreeBSD 6.x you are encouraged to use portsnap for upgrading the ports tree (instead of cvsup). You still will have to use portupgrade to install the new ports you'll have downloaded.

- # portsnap fetch (this downloads a new ports snapshot)
- # portsnap install (create a tree from the snapshot downloaded)

- Once you've done that the first time, next time you need to upgrade your ports do:

- # portsnap fetch
- # portsnap update