# Purpose

1. Learn to use `lpq` and `lprm` commands.
2. Learn to manage your file space. Restrict size of Netscape disk cache.
    - `quota -v`, `du` commands
3. Using the File Manager to copy and remove files.
4. Explore the `string` Class.
5. Explore the `grid` Class.

# Make Printing Easier

During this course you will frequently have to print files either for your own studying or for handing in. There are problems which can occur associated

with printing and it is important that you understand both the problems and how to resolve them. In this lab we will discuss two such problems.

1. Occasionally, you may print something and discover that you printed the wrong file. You can inspect a printer's queue by issuing the command `lpq` and remove a print job by using the `lprm` command and the job number listed in the `lpq` command. Here is a sample run.

```
altair{1077}% print test5.cc
[test5.cc (C++): 1 page on 1 sheet]
[Total: 1 page on 1 sheet] sent to the standard output
request id is dana350-972 (1 file)
altair{1078}% lpq
dana231 is ready and printing
Rank    Owner   Job     File(s)                 Total Size
active  hyde    972     standard input          14532 bytes
altair{1079}% lprm 972
dana350-972: cancelled
altair{1080}%
```

It is important to know that only the person who submitted the print job can cancel it.

2. Sometimes you might print a job and then find that it hasn't appeared in the printer's output tray. Using the

# File Space Problems

Each student in this course is given a 20 megabyte file space limit. Though this should be plenty of space for this course, there are things which can happen which can gobble up the space very quickly. Here are two things you can do to help alleviate the problems.

## Resetting the Size of Netscape Cache

**Note:** If you have already set your Netscape cache to a small value then skip these steps.

Make the following modification to the setup of Netscape.

1. Start Netscape, if you haven't done so already.

2. Pull down the Netscape **Edit** menu and select **Preferences** from the menu — the last menu item.

3. A window will open up with a list of items on the left and more stuff on the right. On the left side select the bottom item **Advanced** by clicking on the right-pointing arrow at the beginning of the line. This should bring up two or three new lines below.

4. From the new lines, click on **Cache**.

5. New entries will appear on the right. Change the entry for **Disk cache** to read 500 rather than 5000.

6. Click on **OK**.

You have reset Netscape to maintain a 500K byte cache rather than a 5 megabyte cache.

## How Much Space Are You Using?

Another thing you can do to keep track of your disk space usage is to run the UNIX command

```
quota -v
```

each week. If you execute this command in your home directory it will inform you of the total disk space you are currently using. If the command reports 10,240, then you currently using 10 megabytes — half your quota. If this number gets close to 20,000 then talk to your instructor about how to reduce your file space utilization.

If you want to know how much space you are using per subdirectory use the UNIX command

```
du
```

This will show how the space is being used and what you might like to remove.

# Dragging Files in the File Manager

If the file manager is not currently running then start it by selecting “File Manager” on the “Files” submenu of the Workspace Menu. One of the nicer features of the File Manager allows you to copy files without using the UNIX command prompt. Be sure that the File Manager is open to your home directory. Enter your `Labs` directory to get access to the directories for the labs already completed — the window should now have, at least, directories `Lab1`, `Lab2` and `Lab3` (names may vary). Place the pointer on the folder for `Lab1`, press the left mouse button and keep it down. Now, with the mouse button still pressed, move the mouse so that the folder icon is moved onto the main workspace background — release the mouse button. The icon should now be on the workspace. We call this just-completed action “dragging” an item from one location to another.

## Copying files

Double click the new folder icon which you have placed on the workspace. You should see the icon replaced by a new File Manager window, but this one open to the `Lab1` directory. In the original File Manager window enter the `Lab2` directory. In the `Lab2` directory select a file icon (try `error.cc`) and drag it to the new File Manager window — stopping and releasing the mouse button when the icon is over the main File Manager window. The file should have been moved from the first directory to the second. Repeat the process in the reverse direction.

Now repeat the action with the following modification: before pressing the mouse button (to start the dragging) press and hold down the control key — now press the mouse button and drag the file dropping it on the other File Manager window — this time the file should have been copied rather than moved — i.e., there are now two copies of the file, whereas with a move there will remain just one.

## Clean Up Your Directory Space

Set the two file manager windows as follows. Have one positioned in your home directory and the other in your `Labs` directory. Now look at the files you have in your home directory. If you have files which belong in one of your lab directories then move them there (using "drag and drop"). Also, delete any files which are no longer needed. Especially you should remove all files with an `.exe` extension. If you ever need anyone one of them, you can easily just recompile them.

You may close the second File Manager window.

# The `string` Class

Copy the file `secret.cc` from the `~cs203/Labs/Lab4` directory (the usual place) to your `Lab4` directory (create the directory first if you haven't done so already). This program will ask you to type five words and then five integer values. The idea is that each integer value specifies a character position in the corresponding input word. The program will print out a word

composed of the five characters specified in the input.

Look at the end of this section to see what will be required for your `handin.txt` file.

Compile and run the program — enter the following data (as shown).

```
cheap energy can cause problems
4 2 1 0 5
```

Notice the clever result.

1. Modify the program so that the integer entered is interpreted as the position from the *end* of the string. You must do some arithmetic in order to get the correct result. For the example above, the program output should look like

```
Enter five words: cheap energy can cause problems
Enter five integers: 0 3 1 4 2
Secret word is 'peace'
```

2. Add a new string variable to the program and assign it the value obtained by concatenating together all five words (plus intervening spaces — `word1 + " " +...`). Display the value of this new variable **and** the value obtained by taking the following substring (use `substr`): the whole string without the first two and last two characters.

   The problem, of course, is that you don't know how long the string will be each time — this must work for all input. When you display the value of the substring, display it surrounded by single quote marks such as shown. If the program is run with the five words indicated in the earlier input example, the complete program output should be as follows.

```
Enter five words: cheap energy can cause problems
Enter five integers: 0 3 1 4 2
Secret word is 'peace'
'cheap energy can cause problems'
'eap energy can cause proble'
```

You should be able to make this modification to your program using the `string` class' `substr()` and `length()` methods.

For this section, hand in the final program (since it includes the previous parts) and the output from a single execution.

# The `grid` Class

Look at the end to see what will be required for your `handin.txt` file.

You are to use the `grid` class discussed in Mercer's text on pages 123–129. You should read these pages carefully.

You will need to use the following line to include the class:

```
#include "/home/hydra/COURSES/cs203/include/grid"
```

1. Here is a grid pattern. Write a C++ program which creates and displays this grid.

```
............
....O.......
...####.....
.....O......
..<.........
............
```

2. Add code to your grid program which will cause the player to traverse the grid "picking up" the `O`s, avoiding the `#`'s and returning to the starting place.

   Display the state of the grid initially and when the player has returned to its starting point.

   Consult the Mercer text if necessary.

## What to Hand In

Put in your `handin.txt` file the following:

1. For `string` class section, hand in the final program (since it includes both parts) and the output from a single execution.

2. For `grid` class section, hand in the final program and the output from an execution.