

**CSC112**                      **Spring 2011**  
**Fundamentals of Computer Science**  
**Lab 4**

In this lab, we will add backup capabilities to our Ubuntu/Linux working environment, and we will continue our study of the C++ language.

**Adding backup to your Ubuntu/Linux working environment**

Start your Linux/Ubuntu virtual machine and select full screen mode. Start the firefox web browser in your Linux/Ubuntu virtual machine, and point it to:

`http://menehune.opt.wfu.edu/CSC112`

**Do not use your windows browser for this step!**

You will see one link for downloading:

- `rback` (Linux/Ubuntu 32-bit executable)

**Getting “rback”**

Click on “`rback` (Linux/Ubuntu 32-bit executable)” with the left mouse button. A dialog box will appear. Select “Save File”.

**Installing “rback”**

1. Change to the directory containing the downloaded files. Use the commands:

```
cd ~/Downloads
ls
```

You should see your downloaded files, including:

```
rback
```

Now, we need to ensure that the downloaded file `rback` has the correct execute permissions. Use the command:

```
chmod ugo+x rback
```

We now copy the `rback` program to `/usr/local/bin`. You need to use `sudo` because of ownership and permissions in `/usr/local/bin`.

```
sudo cp -p rback /usr/local/bin
```

## Putting “rback” in the menu

Using the pull-down menus in the upper left portion of your Ubuntu screen, go to:

```
System-->Preferences-->Main Menu
```

Make sure “Applications” is highlighted in the left column and select “New Item” from the upper right region of your Main Menu window. A window will appear with three text-entry boxes. Fill them in as shown below and click “Ok”.

```
Name:    rback
Command: /usr/local/bin/rback
Comment: Rsync backup GUI
```

Close the “Main Menu” window. You should now be able to start `rback` from the “Applications” menu.

## Starting “rback” automatically when you login

Using the pull-down menus in the upper left portion of your Ubuntu screen, go to:

```
System-->Preferences-->Startup Applications
```

A window will appear. Click on the **Add** button on the right side of the “Startup Applications Preferences” window. Yet another window will pop up with three text-entry boxes. Fill the text boxes as shown below and click “Add”.

```
Name:    rback
Command: /usr/local/bin/rback
Comment: Rsync backup GUI
```

## Completing Setup of rback

The GUI front end (`rback`) starts a shell script which does the remote synchronization (`rsync`) with our two servers: *tristan.cs.wfu.edu* and *isolde.cs.wfu.edu*. The backup shell scripts are customized for each student. There are also two small configuration files which control which backup server(s) to use, and the time (in seconds) between backups. A tar file is provided for you to simplify the rest of the setup.

The first step is to download the tar file into your home directory using `sftp` to *telesto.cs.wfu.edu*. The file is named using your WFU (e-mail) name with the addition of the `.tar` filename extension.

```
cd
sftp telesto.cs.wfu.edu
> get your_email_name.tar
> quit
tar xf your_email_name.tar
```

Naturally, you will use your real WFU on-line name in place of “`your_email_name`” in the commands shown above.

## Continuing with C++

Create a directory named Lab4 and keep all of your source and compiled programs in the directory Lab4.

In this Lab, we will learn to use the bits in an unsigned integer to represent sets. In this case, we will represent a set of lower case letters. The low order bit represents the letter 'a'; the next higher order bit represents the letter 'b', and so forth down to 'z'. There are 32 bits (4 bytes) in a C++ type `unsigned int`, so there are more than enough bits to represent all 26 letters of the alphabet. For example, the number 3, in 32-bit binary:

```
00000000000000000000000000000011
```

represents the set { 'a', 'b' }. The number 41, in 32-bit binary:

```
0000000000000000000000000000101001
```

represents the set { 'a', 'd', 'f' }.

1. Write a C++ program that reads in two numbers, and stores those numbers as type “unsigned int”. Your program should then:

- (a) Print the sets corresponding to the two numbers. Your output should use set notation, e.g.,

```
{ 'a', 'd', 'f' }
```

- (b) Use a bit-wise logical operator to compute the set union of the two sets. Print your result.
- (c) Use a bit-wise logical operator to compute the set intersection of the two sets. Print your result.
- (d) Use a bit-wise logical operators to compute the set difference. Print your result. Note: Given two sets  $A$  and  $B$ , the set difference, denoted  $A - B$  is defined as all the elements that are in  $A$ , but not in  $B$ .

Test your program with input 53208257 and 8904049. You should get the sets:

```
{ a g h k n o p q r t v y z }    and    { a e f g i k l m o p q r s x }
```

2. Write a C++ program that reads a sequence of lower case letters (up to the end-of-line) and computes the numerical representation (unsigned int described above) of the set. Output the unsigned int as a number. Your program should check that the input sequence consists of ONLY lower case letters. If some other character other than 'a' through 'z' is found before the end of line, your program should print an error message and exit. Use your program from part 1 to verify that this part is working correctly.

**Turn in:** Change to the directory containing the sub-directory “Lab4” Create a file named “lab4.tar” using the command:

```
tar cf lab4.tar Lab4
```

Upload the file “lab4.tar” to your account on telesto.