COP4530 – Data Structures, Algorithms and Generic Programming
Recitation 2
Date: September 5th 2005

Lab objectives:

1) **Review namespace usage.**
2) **Review of Command-line Input and File I/O**
3) **Review program compilation.**
4) **Review of the make utility and `makefile`**
5) **Learn how to submit projects using the submission scripts.**

Setup tasks:

1. Logon to your CS account.

2. Create a directory named **`cop4530`**.

3. Go into the **`cop4530`** directory and create a sub-directory named **`recitation`**.

4. Go into the **`recitation`** directory. Type the command **`pwd`**. You should see something similar to the following on the screen:

   For undergraduates:
   `/home/majors/your_username/cop4530/recitation`

   For graduates:
   `/home/grads/your_username/cop4530/recitation`

   For non-CS majors:
   `/home/class/your_username/cop4530/recitation`

5. Type the following command,

   `cp -r ~cop4530/fall05/recitation/rect2/ .`

   Note: You should see the following error messages,

   ```
   cp: cannot open
   `/home/courses/cop4530/fall05/recitation/rect2/makeutil/exercise/makefile' for
   reading: Permission denied
   ```

   You may ignore the error message as the file **`/home/courses/cop4530/fall05/recitation/rect2/makeutil/exercise/makefile`** is a solution to your exercise and will only be made available at a later date.

Task 1 : Review namespace usage.

1. Go into the **`usenamespace`** directory. Compile the file ***namespace.cpp***. You should see the following error messages.

```
namespace.cpp: In function `int main()':
namespace.cpp:23: error: `cout' undeclared (first use this function)
namespace.cpp:23: error: (Each undeclared identifier is reported only
once for each function it appears in.)
namespace.cpp:23: error: `endl' undeclared (first use this function)
```

   a. Why does this occur?

   b. How do you fix it?

2. Read the code and write down what the output should be.

```
Line1  =

Line2  =

Line3  =

Line4  =

Line5  =
```

3. Compile the program and run the executable. Observe the output and determine if your answers were correct.

Task 2: Review of Command-line Input and File I/O

1. Go into the **`fileio`** directory. Open the file **`main.cpp`** and study the code.

```
#include <iostream>
#include <vector>
#include <fstream>
#include <string>

using namespace std;

int main(int argc, char **argv)
{
  if (argc < 2)
    {
      cout << "Usage error : " << argv[0] << " <filename>" << endl;
      exit(1);
    }

  char filen[256];
  strcpy(filen, argv[1]);

  ifstream infile;
  infile.open(filen);
  if(!infile)
    {
      cout << "Error: Could not find/open file" <<endl;
      exit(1);
    }

  string name;
  vector <string> V;

  //Reading in from file
  while (infile >> name)
    V.push_back(name);

  //Remember to close your file after reading
  infile.close();

  for (int i = 0; i < V.size(); i++)
    cout << "V[" << i << "] = " << V[i] << endl;

  return 0;
}
```

Note:

1. The program above needs the **`vector`**, **`fstream`** and **`string`** libraries.
2. The program reads in a filename passed in from the command line, reads from the file one item at a time as strings, and adds the read string items into a vector.
3. Finally, the program displays the contents of the vector onto the screen.

2. Compile the file **`main.cpp`** using the command,

```
g++ main.cpp
```

3. Run the executable by typing,

```
./a.out namelist
```

Exercise

Write a program that will read in a list of integers from a file. Add the values of the integers and print out the sum of all integers read.

Task 3 : Review program compilation.

1. Go into the **`makeutil`** directory. Open the file **`main.cpp`** and study the code.

```
#include <iostream>
#include <print.h>
#include <largest.h>

int main()
{
  const int ASize = 8;
  int A[ASize] = {32, 4, 8, 62, 3, 42, 23, 9};
  PrintArray(A, ASize);
  GetLargest(A, ASize);

  return 0;
}
```

2. Compile the program ***main.cpp*** by typing

   `g++ main.cpp`

   and observe the error message. You should see something similar to the following:

```
main.cpp:9:19: print.h: No such file or directory
main.cpp:10:21: largest.h: No such file or directory
main.cpp: In function `int main()':
main.cpp:16: error: `PrintArray' undeclared (first use this function)
main.cpp:16: error: (Each undeclared identifier is reported only once
for each function it appears in.)
main.cpp:17: error: `GetLargest' undeclared (first use this function)
```

   a. Why did this occur?

   b. How do we solve the problem?

Task 4 : Review of the make utility and makefile.

Review 1: Simple `makefile` with dependencies.

```
all: main.x

main.x: largest.o print.o main.o
< TAB  >g++ -Wall -pedantic -o main.x print.o largest.o main.o

largest.o: ./largest.h ./largest.cpp
< TAB  >g++ -Wall -pedantic -c -I. ./largest.cpp

print.o: ./print.h ./print.cpp
< TAB  >g++ -Wall -pedantic -c -I. ./print.cpp

main.o: ./main.cpp
< TAB  >g++ -Wall -pedantic -c -I. ./main.cpp

clean:
< TAB  >rm -f *.o *~ *.x
```

Review 2: `Makefile` with macros.

```
HOME = /home/courses/cop4530/fall05/recitation
CC   = g++ -Wall -pedantic
PROJ = $(HOME)/rect2/makeutil
INCL = -I$(PROJ)

all: main.x

main.x: largest.o print.o main.o
< TAB  >$(CC) -o main.x print.o largest.o main.o

largest.o: $(PROJ)/largest.h $(PROJ)/largest.cpp
< TAB  >$(CC) -c $(INCL) $(PROJ)/largest.cpp

print.o: $(PROJ)/print.h $(PROJ)/print.cpp
< TAB  >$(CC) -c $(INCL) $(PROJ)/print.cpp

main.o: $(PROJ)/main.cpp
< TAB  >$(CC) -c $(INCL) $(PROJ)/main.cpp

clean:
< TAB  >rm -f *.o *~ *.x
```

Note: The **< TAB >** indicators are literal TAB spaces inserted by pressing the TAB key on your keyboard. It is followed immediately by the command you wish to executed, e.g. `g++ . . .`

1. While still in the **makeutil** directory, open the file called **makefile** and observe its content.

```
# Filename    : makefile
# Date        : September 5th, 2005
# Description : Simple makefile example
#

all: main.x

main.x: largest.o print.o main.o
        g++ -Wall -pedantic -o main.x print.o largest.o main.o

largest.o: ./largest.h ./largest.cpp
        g++ -Wall -pedantic -c -I. ./largest.cpp

print.o: ./print.h ./print.cpp
        g++ -Wall -pedantic -c -I. ./print.cpp

main.o: ./main.cpp
        g++ -Wall -pedantic -c -I. ./main.cpp

clean:
        rm -f *.o *~ *.x
```

2. Compile the main.cpp program using the make utility by typing,

   make

3. Run the executable **main.x**. The output should be similar to that of ***Task 2***.

4. Change the name of your **makefile** to **newmakefile** by typing the command,

   mv makefile newmakefile

5. Next, type **make**. You should see the following error:

   make: * No targets specified and no makefile found. Stop.**

   a. Why did this occur?

   b. How do we get around the problem?

Exercise

Write a **makefile** that will compile the program in file **database.cpp** located in the directory **/rect2/makeutil/exercise/**

Task 5 : Learn how to submit projects using the submission scripts.

1. Go into the **`projsubmit`** directory.

2. Copy the submission script from the class account by typing the following command,

   ```
   cp ~cop4530/fall05/submitscripts/rect2submit.sh .
   ```

   Note: This step copies the submission script named **`rect2submit.sh`** into your current directory. Submission scripts collect relevant files from your current directory and send them via e-mail to the class account. This means that you have to have either **`pine`** or **`elm`** activated in your account prior to using the submission script.

   All future submission scripts for projects (unless otherwise specified) shall be obtained in the same manner. The path **`~cop4530/fall05/submitscripts/`** for all submission scripts will remain the same. Only the actual version number (**`projXsubmit.sh`**) of the script will change accordingly.

3. Because the submission script needs an e-mail agent to send the files, you will need to log on to **`shell.cs.fsu.edu.`** You may do this by typing the following in your current directory,

   ```
   ssh shell –l your_username
   ```

   Continue, by typing in the password to your CS account.

4. Go back into the **`projsubmit`** directory. Execute the submission script by typing,

   ```
   ./rect2submit.sh
   ```

5. You should see something similar to the following,

   ```
   Changing directory to ....
   Archiving...
   makefile
   main.cpp
   Sending mail...
   Cleaning up...
   ```

6. Check your mailbox. You should receive the following 2 e-mails.

   i. A confirmation receipt of your submission.

   ii. A copy of all the files submitted.

7. If you do not receive any confirmation, email the TA at toh@cs.fsu.edu for immediate notification of the error. This is done to avoid point deduction for late submission penalty due to system error.

References

Topic	Recommendations
Namespace usage	Url: http://www.glenmccl.com/ns_comp.htm
Command-line Input	Url: http://www.phon.ucl.ac.uk/courses/spsci/abc/lesson11.htm
File I/O	Url: http://www.cplusplus.com/doc/tutorial/tut6-1.html
Make Utility	Url: http://developers.sun.com/solaris/articles/make_utility.html Books: **Managing Projects With make** by Andrew Oram, Steve Talbott ISBN: 0937175900