Part I: This part is for students new to CS Department's turing/hopper system. You can skip this part if you have already accessed your account before.

Follow the link below to activate your Linux account and change your password:

http://www.cs.niu.edu/resources/linux-access.html

Part II: Creating an assignment directory

1. Create a directory for CSCI 340. At the UNIX prompt, type the following command and press [`enter`]:

   ```
   mkdir 340
   ```

2. Change the access permissions for your new assignment directory by executing the following command:

   ```
   chmod 700 340
   ```

3. Navigate to the `340` directory by executing the following command:

   ```
   cd 340
   ```

Part III: Create and execute a C++ program

1. Create a new directory to hold the files for your Assignment 0 program. Change your working directory to that new directory.

   ```
   mkdir assign0
   cd assign0
   ```

2. Open a text editor and create a file called `linkedlist.h`. For beginners, the easiest editor to use would be **nano**. To open the editor and create the new file, execute the command

   ```
   nano linkedlist.h
   ```

   Then type in the following header file:

```
#ifndef LINKEDLIST_H
#define LINKEDLIST_H

class Node {
    friend class LinkedList;
    int data;
    Node* next;
    public:
        Node(int v, Node* n) { data = v; next = n; }
};

class LinkedList {
    private:
        Node* root;
        void reverse_print( const Node* n ) const;
    public:
        LinkedList() { root = NULL; }
        void reverse_print() const;
        void add(int val);
};

#endif
```

Save the file by executing `Ctrl-O`.

Exit **nano** by executing `Ctrl-X`.

3. Create and edit the implementation file `linkedlist.cc` using the text editor.

```
#include <iostream>
#include "linkedlist.h"
using namespace std;

void LinkedList::reverse_print( const Node* n ) const {
    if ( n == NULL )
        return;
    reverse_print(n->next);
    cout << n->data << ' ';
}

void LinkedList::reverse_print() const {
    cout << endl;
    reverse_print( root );
    cout << endl;
}

void LinkedList::add(int val) {
    if ( root == NULL )
        root = new Node(val, NULL);
    else {
        Node* n = new Node(val, root);
        root = n;
    }
}

int main( ) {
    LinkedList LL;
    LL.add(3);
    LL.add(5);
    LL.add(8);
    LL.add(4);
    LL.add(9);

    LL.reverse_print();

    return 0;
}
```

4. Compile and link the program by executing the command

```
g++ -g –Wall linkedlist.cc –o linkedlist
```

5. Correct any typing errors if necessary and re-compile.

6. You can run your program by executing the command

```
./linkedlist
```

7. The correct running result is: `3 5 8 4 9`

Part IV: Creating and using a `Makefile`

1. Create and edit the `Makefile` using the text editor.
   ```
   nano Makefile
   ```

2. Type in the following: (NOTE: the lines that begin with `"$(CC)"`, `"./"` and `"- rm"` **MUST be indented by using a tab character, not spaces**.)

```
CC = g++
CCFLAGS = -Wall -g

linkedlist: linkedlist.o
	$(CC) -o linkedlist $(CCFLAGS) linkedlist.o

linkedlist.o: linkedlist.h linkedlist.cc
	$(CC) -c $(CCFLAGS) linkedlist.cc

run:
	./linkedlist

clean:
	-rm linkedlist.o linkedlist
```

3. Compile and link the program by executing the command
   ```
   make
   ```

4. Correct any typing errors in `Makefile` (pay attention to the tab character), then run `make` again.

5. You can run the program by typing:
   ```
   make run
   ```

6. The correct program output is: `3 5 8 4 9`

7. You can remove object files and executable file by typing the command:
   ```
   make clean
   ```

Part V: Practicing basic UNIX commands.

Follow the link below to browse and learn basic UNIX commands. Try executing the commands. Practice at minimum the commands of `ls`, `mkdir`, `pwd`, `cd`, `rm`, `mv`, `cp`, `less`, `chmod`, etc. at this time. You will need to learn more commands such as `cat`, `ftp`, `diff`, `i/o` redirection during this semester. Ask the instructor or TA if you have a doubt.

http://faculty.cs.niu.edu/~mcmahon/CS241/c241man/node8.html