Intermediate/Advanced Computer Programming
Review

• What is the difference between UNIX and Linux?
• Name/describe the three basic layers of a typical OS
• What is the OS kernel?
• Which memory space do typical C++ applications execute?
• Name some UNIX commands
• Name/describe the steps to create an executable from C++ source code
Unix

- Single UNIX Specification defined by the Open Group
- Standard definition for the Unix system API
  - Specifies requirements for a UNIX system
- Not code
  - Allows different implementations
- Contains POSIX specifications
- Defines system interfaces, headers, commands, and utilities
- http://pubs.opengroup.org/onlinepubs/009695399/
Building an Executable

Source Code

Preprocessor

Preprocessed Source Code

Compiler

Assembly Source Code

Assembler

Object Code

Linker

Executable

Library
Building an Executable

• Machine code
  – Language directly understandable by the hardware (e.g. 1's and 0's)

• Assembly code
  – Human readable machine code

• Object code
  – Separate portions of machine code that have not been linked into a complete executable

• Executable
  – Encoded instructions to perform a task
Building an Executable

class.cpp

main.cpp

class.o

main.o

a.out
Unix Commands

• *ls*
  – List directory contents
• *mkdir*
  – Make directories
• *man*
  – Online manual pages
• *cat*
  – Output file contents
Redirecting/Piping

Text terminal

- Keyboard
  - #0 stdin
- Display
  - #1 stdout
  - #2 stderr

Program
Redirecting/Piping

- Streams/channels
  - flow of input and output between a program and its environment
  - Standard streams
    - Initial connections when program begins

- **command > file**
  - Output redirected to file
  - File is truncated if it already exists, otherwise file is created

- **command >> file**
  - Redirect output to file
  - Append output to file if it already exists, otherwise create file

- **command < file**
  - Contents of file are used as input to command (rather than keyboard)
Effective Programming
Process to Build Program

1) Define the problem
   - Specifications
   - Examples

1) Design the program
   - Diagrams
   - Pseudo-code

1) Write the source code (e.g., functions, classes)

2) Test the program

3) Repeat steps 1-4 until desired program is verified/created
Testing

• Ensure that program performs correctly
• Exhaustive testing
  • For all inputs, do we get the correct output
  • Time consuming
  • Impractical
• Test boundary conditions
  • Zero
  • Largest number
  • Smallest number
  • Positive numbers
  • Negative numbers
Process to Coding

1) Comment
2) Code
3) Compile
4) Execute
Process to Coding

1) Comment
2) THINK
3) Code
4) THINK
5) Compile
6) Execute
Code Evaluation

• Correctness

• Quality of Code
  • Readability
  • Conveying program semantics to the compiler
  • Scalability
  • Efficiency
Comments

A program serves two masters.
- Code tells the computer what to do.
- Comments describe what the program does to the poor programmer who has to maintain it.

There are two types of comments in C++.

// Comments that begin with double-slash
// and go to the end of line

/* Comments that start with slash/star */
/* and go to star/slash */
/*
 * The second version can be used
 * for multi-line comments
 */
Precedence Rules

1. ( ) [ ] -> .
2. ! ~ ++ -- (type) - (unary) * (dereference) & (address of) sizeof
3. * (multiply) / %
4. + -
5. << >>
6. < <= > >=
7. == !=
8. & (bitwise and)
9. ^
10. |
11. &&
12. ||
13. ?:
14. = += -= etc.
15. ,
Practical Precedence Rules

1. * (multiply) / %
2. + -

- Put parentheses around everything else.
Initializing Variables

• Variable = value;
  – int var = 5;

• C++ style
  – int var(5);

• Arrays
  – int temperature[4] = {100, 95, 92, 98};
  – int temperature[] = {100, 95, 92, 98};
  – First element is temp[0]
  – Last element is temp[3]
# C++ Strings

Bring in the string package using the statement:
#include <string>

// Declaring a string
    string my_name; // The name of the user
Assigning the string a value:
    my_name = "Student";
Using the "+" operator to concatenate strings:
    first_name = "student"; last_name = "student";
    full_name = first_name + " " + last_name;
Arrays

• Bounds
  - int temperature[4] = {100, 95, 92, 98};
  - int tmp = temperature[3];
  - int tmp = temperature[55];
  - temperature[55] = 85;

• Multi-dimensional arrays
  - type variable[size1][size2];
  
  int matrix[2][4] =
  {
    {1,2,3,4},
    {10,20,30,40}
  };

**if Statement**

- General form:
  
  ```
  if (condition)  
  statement;
  ```

- If the condition is true (non-zero), the statement is executed.
- If the condition is false (zero), the statement is not executed.

- Example:
  ```
  if (total_owed <= 0)  
  std::cout << "You owe nothing.\n";
  ```
Loops

• Sequence of statements that can be executed more than once in succession
  
  ```
  for (initializer ; condition ; iteration statement)
  body-statement;
  ```
  
• How can we write an equivalent while loop?
Loops

for (initialization; condition; iteration statement) {
  body-statement;

  initialization;

while (condition) {
  body-statement;

  iteration statement;
}