CS 138 - Data Abstraction and Implementation

Kevin Carruthers

Winter 2013

Unix and the Unix Shell

Unix and C were developed in the 1970s at Bell Labs / AT&T in New Jersey by Thompson, Ritchie, Kernighan, and Pike. Later, C++ was also developed there by Stroustroup et al.

BSD came from UC-Berkeley. Free/Net/OpenBSD forked from this. NeXTSTEP is based on this, MAC OSX is based on NeXTSTEP.

Solaris/SunOS was developed by Sun Microsystems in the 1980s and is now owned by Oracle.

There exist AIX, HP-UX, SCO, and other commerical Unices.

Linux (1990s, open-source). Android and KindleOS are based on this.

BB10 is based on QNIX.

GNU/Linux

GNU utilities run on top of Unix. Some are re-implementations of Unix tools, but GNU's Not Unix. Linux is technically just the kernel, but is mostly useless without a shell, compiler, CL utilities... thus a Linux distribution contains mostly GNU free software, and should properly be termed GNU/Linux.

The Shell

sh is the Bourne Shell, the first practical, widely-used shell. Later implementations include bash, ksh, csh, tcsh.

Filesystem

Pretty much all OSs have a file system, though you can't see it in iOS. Your home directory is where your personal stuff is stored. You can reference it by a ~. For example, user 'god' has a home directory ~god.

Some commands in Unix

```
$ pwd
$ cd
$ ls (-laFt)
$ g++ -o test test.cc
$ ./test
$ echo
$ cat
$ less
$ make
$ cp/mv/rm (-iv)
$ (s)diff x y
$ find [dir] (-name) (-type f | d) (-not | -a | -o expr) patt
$ grep [-irnvEF] pattern-string file-list
$ sort (-n | -c)
```

or more generally

```
$ cmd -opt1 -opt2 arg1 arg2 arg3
```

Commands may be built into the shell, external, or aliases. The shell looks for aliases, then built-ins, then externals. External programs are found by a predefined path (/usr/bin, etc)

What does the current command do in local context?

```
$ which cmd
```

File Permissions

Each file or directory has three sets of permissions: user, group, and other. Groups are arbitrary collections of userIDs defined under admin control. A user can belong to multiple groups.

The three types of permissions are read (4), write (2), and execute (1). For directories, read means "see what files are inside", write means "add/delete files inside", execute means "cd into it".

```
$ chmod (-r) permissionsChanges nameOfFileOrFolder
```

Use ugo(a) and +/- or octal

```
$ ssh (-Y) (-1 userID) (user@)hostname
```

Globbing

- $*$ matches zero or more characters
- ? matches one character
- , matches any alternative in the set
- [] matches one character in the set
- ! not
- $-$ creates ranges. Escape it by putting it at the end or beginning of the set
- $*$ doesn't match dotfiles
- single quotes protect everything, double quotes protect everything except doublequotes, backquotes, variables.

IO Redirection

- $<$ means take input from file
- $>$ means overwrite output to file
- $>>$ means append output to file
- $|$ means pipe the first output into the second input
- $2 > \&1$ means pipe stderr to stdout

C++

```
// hi mom!
using namespace std;
#include <string>
#include <iostream>

const string kidDrinks = "juice";
string adultDrink = "coffee";

int main(int age; char argv[]) {
  int age = 100;
  while(age > 0) {
    cout << "How old are you?";
    cin >> age;
    cout << "Would you like some ";
    if(age < 18) {
```

```
    cout << kidDrink;
  } else {
    cout << adultDrink;
  }
  cout << "?" << endl;
  if(age == 49) {
    adultDrink = "La chouffe";
  }
  return 0;
}
```

Boolean Type

Boolean types have two special cases, true and false.

```
bool done = false;
while(!done) {
  // do stuff
  if( ... ) {
    done = true;
  }
}
```

You can also use numbers as boolean values: zero means false, other values mean true.

```
if(n)
  (n == 0)
  (p = NULL)
  (NULL == p)
```

C++ Strings

Much nicer, more convenient, and safer than char*. We only use char when dealing with legacy C compatibility. Need to include string.

```
#include
int main( ... ) {
  string fruit = "apple";
  string tree = "pine";
  cout << tree + fruit << endl;
  string f = tree + fruit;
  if(f == "pineapple") {
    cout << "yup, same";
  }
  int n = f.length();
  cout << "Beyond the " << f[0] << f.at(4) << f.substr(n-2) << endl;
  cout << "Beyond the" << f[0] + f.at(4) << endl;
}
```

Types

```
cout << "n" << endl;                  \\ n
        'n'                           \\ n
        (int)'n'                      \\ 109
        (string)"m" + (string)"g"     \\ mg
        (string"m" + 'g'              \\ mg
        'm' + 'g'                     \\ 212
        "m" + "g"                     \\ error
```

CLI

```
int main(int argc, char* argv[]) {
  cout << argc << endl;
  for(int i = 0; i < argc; i++) {
    cout << "arg " << i << " is" << argv[i] << endl;
  }
  string s = argv[1];
  cout << s + "hi mom" << endl;
  \\ argv[2] = s;                       This is wrong and stupid
  argv[2] = (char *) s.c_str();
}
```

IO

Sample IO

```
#include <iostream>
```

gives cin, cout, cerr: three special variables you can do IO with.

IO overloads the double arrow operators.

Input and Whitespace

```
cin >> foo >> bar;
```

gives the first two tokens, ignoring all whitespace.

```
string line;
getline(cin, line);
```

gives the entire line of input, newlines are removed.

Input and EOF

```
eof()  // true is EOF
fail() // true if EOF
```

both can be used with other streams, and neither will trigger until you go too far.

Example:

```
int main(...) {
  double sum = 0;
  int count = 0;
  while(true) {
    double next;
    cin >> next;
    if(cin.fail()) {
      break;
    }
    sum += next;
    count++;
  }
}
```

Note that we can also use

```
if(!cin)
```

Or more concisely

```
double next;
while(cin >> next) {
  sum += next;
  count++;
}
```

File IO

ifstream and ofstream are used for file IO. We can't use the filename directly, so we associate it with a stream.

```
ifstream ifstr("foo");
```

Stream CXRS expects char* not strings.

Note that we need to immedaitely check whether stream creation was successful.

```
#include <iostream>
#include <fstream>

int main(int agrc; char* argv[]) {
  if(argc <= 1) {
    cerr << "Error" << endl;
    exit(1);
  }
  ifstream is_raw_grades(argv[1]);
  if(!is_raw_grades) {
    cerr << "Error" << endl;
    exit(2);
  }
  ofstream pass("passes");
  ofstream fail("fails");
  // should check errors here

  int grade;
  string name;
  while(is_raw_grades >> grade >> names) {
    if(grade >= 50) {
      pass << grade << " " << name << endl;
    } else {
      fail << grade << " " << name << endl;
    }
  }

  is_raw_grades.close();
  pass.close();
  fail.close();

  return 0;
}
```

Note that

```
ifstream in("input.txt");
```

is alright, but

```
string mFilename = "input.txt"
ifstream in(mFilename);
```

is not, because streams can be opened only with char* but not string.

There are two ways to open files:

1. Call the cxr (ifstream in(”foo”);)
2. Create stream and connect later (ofstream out; out.open(”bar”);)

Regardless, we must close all sreams with out.close();

```
int main(int argc, char* argv[]) {
  ifstream in1;
  in1.open("in1.txt");
  ifstream in2("in2.txt");
  if(!in1 || !in2) {
    cerr << "ERROR" << endl;
    exit(1);
  }
  ofstream out;
  out.open("out.txt");
  if(!out) {
    cerr << "Error" << endl;
    exit(2);
  }
  string s;
  in1 >> s;
  out << s " to 1" << endl;
  out.close();
  out.open("out2.txt");
  if(!out) {
    // error stuff
  }
  in2 >> s;
  out << s << " to 2" << endl;
  out.close();
  in1.close();
  in2.close();
}
```

cin, cout, cerr

They are global variables from the C++ standard libraries, included in <iostream>. cin is an instance of the C++ class istream, cout and cerr are instances of the C++ class ostream.

Inheritance and Polymorphism

```
void log(ostream output) {
  output << "42" << endl;
}

int main (...) {
  ostream myout("foo");
  log(cerr);
  log(cout);
  log(myout);
}
```

ofstream inherits from ostream, so any aspects of ostream are also aspects of ofstream.

C++ Arrays

Almost exactly like C arrays

```
int A[15];         // OK
const int N = 20;
int B[N];          // OK
int M;
cin >> M;
int C[M];          // Not OK
```

References to arrays are unchecked. We have to pass array.extent(size); to procedures. For example:

```
int findMax(int arr[], int N) {
  int max = 0;
  for(int i = 0; int < N; i++) {
    if(arr[i] > max) {
      max = arr[i];
    }
  }
  return max;
}
```

Vectors

A vector is a container data structure from the C++ STL. It is like a C-Style array except that

- I is generic and type safe
- Index bound checking is possible (optional)
- It can be resized on the fly, either by element (pushback) or by chunk

Other STL data structures are deque, list, map, and set.

Iteration

Simple, normal, numeric approach.

```
for(int i = 0; i < v.sive(); i++)
```

Abstract, powerful, confusing. (STL iterators)

```
for(vector<string>::const_iterator i = v.begin(); i != v.end(); i++)
```