COMP 206

Sept 6

Software Systems- first in line of systems courses

David Meger? Robotics prof

Course objective: ability to build an excellent software system

-interaction with hardware, low level systems programming

-lowest level application code

-develop programs in unix, emphasis on systems programming

-learn to use C, python, bourne-shell scripting, makefiles, version control, and profiling and debugging tools in a unix software dev environment

-Ubuntu 16.04

-emphasis will be on applied techniques to “get the job done”

-how to maintain a codebase

today:

-ensure you can access linux in Trottier

-in person, remotely, browse files, create and edit text docs, use web, etc

-install on your own machine

-find our offices and introduce yourself in the first office hours

-ssh section at cs.mcgill.ca/docs/x

Sept 7

History of unix

1960s- MULTICS project (MIT, Bell, GE, AT&T)

1970s- the idea of UNICS or UNIX takes care of everything involving talking to hardware

1980s- flavours of Unix started being open source- GNU project

1990s- Linus Thorvalds, Linux

Unix systems

SunOS/Solaris – Sun Microsystems

NetBSD/FreeBSD

Unix philosophy

-multiuser/tasking

-toolbox approach

-flexibility/freedom

-conciseness

-everything is a file

-file system has places, processes have life

-designed by programmers for programmers

Unix Structure- the operating system

OS lives at lowest level to interact with hardware

Kernel and system calls between hardware and programs

Kernel- in RAM

-login

-task switching, multi-processing

-basic interface with user and programs

-drivers, run-time stack, heap

OS defines a file system which allows the disk to be used by everything else in a system, provides an interface

Controlling unix- shell- more advanced ui

Has global memory, scriptable to produce complex behavior

Outside of kernel

File system

/ 🡨root

bin (binary) – core unix programs

etc – config files for users/groups

lib- libraries, esp. libc.so

usr- kinda a mirror of bin, less systems-y

dev- device, file handles,

tmp

home- user’s files

commands

important resources:

mycourses

mcgillcomp206fall2016.slack.com

office hours: Mon 1:30pm in MC 112N

unix programs

-shell is the command line interpreter

-is just another program

SHELL COMMAND DAY FUCK YE

-a program or command interacts with the kernel

-may be any of:

-built-in shell command

-interpreted script

-compiled object code file

exiting- ctrl-D is interrupt

exit works too

unix command line structure

man command

man –k command means search

man is kind of old school version from before google

command has format:

command options arguments

arguments indicate on what the command is to perform

pwd print working directory

cd change working directory

mkdir make directory

rmdir remove directory

rm removes a file

ls options

-a lists all files

-l gives long thing- each line lists type field, access permissions- first three user/owner, second 3 assigned unix group, last 3 others

says d if it’s a directory, f it’s a file, l if link, followed by rwx (read write execute)

changing specific permissions

chmod change file or directory access permissions

chgrp change group of he file

chown change owner of a file

u user owning file

chmod u+w [file] gives user write permission

Sept 14

Set of slides labeled 1b or 1c or something with shell commands, learn them

Pipes- holder for a stream of data- can be used to hold the output of one program and feed it to the input of another

Tail- displays end of file, -f will continuously update

>> appends to file, > rewrites

| is pipe

shell variables

shell keeps track of a set of parameter names and values

some of these parameters determine the behavior of the shell

access a shell variable with $ first

no whitespace when defining variables

displaying shell variables- prefix the name of a shell variable with $

echo command will do echo $HOME and $PATH

commonly set path variable to new places, i.e. where custom commands live

set command

PS1 variable

Fancy bash prompts

\t \w…

to always use the same prompt there’s a config file in home directory

~ is placeholder for home directory

~/.bashrc is permanent

PATH is a list of : delimited directories, is a list and a search order

If you follow a command line with & it’ll run in the background, and allow you to do other stuff while the first job is running

fg %[job number] brings it into foreground

ctrl-z puts it in background

ps lists processes run in that terminal

u will list all processes

wildcards (metacharacters) for filename abbrev

when you type in a command line the shell treats some characters as special

\* matches anything i.e. a\* will list anything in a directory that starts with a, and a\*b will list anything that starts with a ends with b

Sept 16

Quoting- “\*” way to change the behavior

\ will escape quotes, i.e. \” prints “

quotes will change meaning of the string- ls “foo fee file?” only searches for a file name with spaces

single quotes will stop special characters from being treated as special

backquotes – string is replaced with the result of running the command in backquotes

Sept 19

Assignment out today

Shell Scripting

Further observations, real coding

Further commands:

Expr- evaluates args as arithmetic, expresses and prints the result

i.e.

$a=3

$b=5

echo `expr “$a” “+” “$b”`

8

double quotes necessary to disable wildcard functions

test- does comparisons on files, boolean result is the exit code of the program

uses options

test –r only takes one argument, returns true if readable file with that name exists

grep searches for a pattern in a longer string- a match is reflected in exit code of the program

minor commands

sleep N, delay N seconds

wc- word count

lines words characters filename

-l only gives lines

top is useful for monitoring, blocks commandline

ps –aux is a one-time process, quits after output

if, while, for

if *program*

then

*commands*

fi

execute the body if the program returns an exit code of 0 (success)

special syntax for running test w/o saying the word text

#!/bin/bash

a=hello

b=hello

if [ $a = $b ] //[] takes the place of test

then

echo Match

else

echo No match

fi

= tests for string equality comparison

bash filename runs as if entered in command line

if

then

elif

else

fi

/dev/null throws stuff away

how to do that thing without creating any junkfiles? (not using grep –q)

Sept 21

Shell review

Head tail- show part of a file

-n 5 file.txt [show the first 5 lines only]

Cut paste join- deal with colums in a text file

Cut has odd syntax, check it

Sort- reorders the lines in a file

-n treats input as a number

Tr-translate characters

Will help remove inconsistencies in file naming

Uniq- finds repeated or unique lines in a file

Sept 26

C intro

Compiled language- compile with gcc

Takes in c source code, results in .out file

Libraries- for printing

#include <stdio.h> //<> for system libraries

int main(int argc, char \*argv[]) //int argc counts args provided- one implicit

//

{

printf( “Hello world.\n”);

printf(“I had %d arguments.\n”, argc);

printf(“the first was “, argv[0]);

return 0;

}

pros/cons

-low level control of execution and memory

-functional paradigm encouraged for clean algorithm dev

-sometimes too much control

-OO thinking has been promoted best practice for large software programs

ImageMagick

-countless useful apps to work with images and even movies

Convert is a program that interacts with its surrounding software systems

-parse cl args

-process files

-utilize system resources

-returns a status

-prints to standard outputs

no classes in C

\* refers to pointers

Sept 28

Built in data types in C

Integer- two’s complement

Floating point- float, double

Boolean- short, int, long- no Bool by default in C. use 1 and 0

Char- 8 bit, almost equiv to unsigned short ints, uses ASCII

String- char\* 32 bit- address in memory, special case of pointer

Data type sizes

Declaring variables in C

SCOPE MODIFIER TYPE VAR\_NAME;

SCOPE MODIFIER TYPE VAR\_NAME = VALUE;

SCOPE MODIFIER TYPE VAR1, VAR2, … , VARn;

Scope: static, extern, or it is not used

Modifier: unsigned, short, long, or not used

Type: one of the built in types

Var\_name: any word not beginning with a number or reserved symbol like + or -, case sensitive

Examples:

long double x; //long may be ignored depending on the compiler

unsigned int;

short double == float

implicit and explicit casting

requires care by programmer to ensure intended use cases

compilers are improving- often exceeding the max size of a short int will trigger an overflow

Sept 30

Bunch of updates:

C99 standard added reserved word for boolean \_Bool

Char is actually -127 to 128, depending on compiler

Short is actually 16 bits, int is 32, long is 64

Pointers are either 32 or 64 bits depending on OS/processor properties

Placement of variables determines scope

Separate declaration from implementation of functions:

Void add(int, int);

Void add(int a, int b)

{

x = a + b;

}

you must declare a function before using it!

Nice programming practice- use local variables as much as possible

If using globals, list them at the top of the program, don’t mask them out locally with identical names, and use static as little as possible

Variable assignment

Just like in java you can assign a value to a variable using the equal sign

In C you can chain assignments

Unlike java variables are not defaulted to 0

Operators

= + - \* / % ++ -- (for those two, order will change before or after contents of the rest of the line) += -= \*= /= %=

Oct 3

C strings are char arrays

Char \*x= “bob”;

Pointers

Give C power and flexibility regarding memory management

Allows dynamic allocation

Declare with \*

Dereference with \*

Char \*x = “abc”;

Char c = \*x;

Oct 5

^

x holds an address

obtain address with &

++ on a pointer will change the addresses pointed to instead

C string concepts

Do not use == to compare

Null-terminate with \0

Each character can be treated as an 8-bit int, which can be powerful or dangerous

Compare with strcmp(lhs, rhs); //reurns 0 if identical

string.h is useful string lib

Oct 12

Basic I/O operators

For stdin- int getchar() – more general cousin – int fgetc()

EOF- end of file is returned by input functions when they can no longer give useful input: for a fixed file this is simple, but to gen from streamed input like keyboard, ctrl-D- in C this typically has value -1

\n: new line

\r: carriage return (mainly not used in Unix)

if something like fgetc() == EOF, error

fread

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream)

ptr- this is the pointer to a block of memory with a min size of size\*nmemb bytes

-size- this is the size in bytes of each elem to be read

-nmemb- this is number of elems, each one with a size of size bytes

-stream- this is pointer to a FILE object that specifies an input stream

(Un)Buffered I-O

your program does not directly access files/keyboard/pipes/etc

C’s file I/O functions make requests to OS through libs and system calls

For efficiency, the default IO mode in Unix is buffered- this speeds up effective operations

-grouping up data to save operations

review slides from Friday!

Oct 14

Midterm:

Mix between mc etc

Mostly about tools we’ve learned

Shell script, bash, c, language features up to next wed

Working with memory

Bit-wise operations

Type casting

Pointers

Shifts:

Bit\_arg<<shift\_arg

Shifts bits of bit\_arg shift\_arg places to the left- equiv to multi by 2^shift\_arg

Same with right

Watch for overflow

Bitwise logic

L & r and

L | r or

L ^ r xor

~arg not

c is “statically typed”

it’s permissive to allowing many type conversions

Oct 17

If type is wider than another type you don’t lose data typically

Questions: double to int- C knows that it was a double, rounds, or goes to ieee standard?

^ came up in class, only if you use pointers

Truncation of long int to int for negative numbers?

Basically, don’t do it

sizeof() returns length of variable or typename

i.e. sizeof(unsigned short int)

arithmetic conversions- automatically promotes to the higher precision type according to the following hierarchy

int

unsigned

long

unsigned long

float

double

long double

type casting necessary same format as java

pointers can reinterpret memory

int \*p; //pointer to int value

int a = 257;

p = &a; //stores address of a into p

char \*p; //pointer to char value

int a = 257;

p = &a; //will store reference to first byte of a

which is first byte? Depends on the system. Great. Big-endianess vs little-endian

it’s convention. Most systems we deal with will be little-endian- the first address will store the least significant bit

back to Steganography

bmp:

start of file has a header

has ascii characters bm

Oct 24

Review!

C I/O functions

Endian

Pointers

Advanced shell programming

Bitwise calculations

Pointers

Ampersand takes a variable and creates a pointer, star gets you back the variable that was being pointed at

Oct 26

Debugging c programs/strategy

Strategy 1:

Printf everything!

Pointer’s addresses, increments, etc… see where stuff goes wrong

Useful, but slow and messes up timing sometimes

Preprocessor

#include is a preprocessor directive

almost all the time #include is just for headers

others are

#define, #ifdef, #pragma

define defines a variable, similar to a global but unchangeable

ifdef if define variable exists

#ifdef DEBUG

#define debug(x,y) printf(x,y)

#else

#define debug(x,y)

#endif

compile with gcc –DDEBUG or not to toggle, or define with headers

gdb debugger

GNU debugger, available almost everywhere gcc is

CLI, some key commands:

run <args>- can tell you where a segfault is

break <line# or function name>

can step through the program with next, nexti

disp <var> to display variable when stepping through

finish

…or just use eclipse. Because fuck emacs

creating and managing larger C programs

multiple-file projects and libraries

most obvious solution: list both main and other program when compiling if calling a function from another program

OR use a header file that calls the other program

#include file.h

save the file.h as the function declaration, when compiling list all c files

this gives you an object file, re-compilable individually by gcc -c

Oct 31

Struct TYPE\_NAME

{

FIELD1; //can declare types before this, i.e. int classSize;

FIELD2; //type VAR; not initialized

} VAR\_NAME; //optional identifier variable

//example declarations

struct TYPE\_NAME var1, var2, array[10], \*p;

struct COURSE cs206;

cs206.numberOfStudent = 60;

strcpy(cs206.nameProfessor, “Alex”);

//or

struct COURSE cs206 = {60, “Alex”};

Dynamic Memory

Malloc and calloc use heap

Heap’s job is runtime configurable requests, explicitly asked for and explicitly released

Malloc and calloc allocate a block of memory and return a pointer to that allocated memory

Number of bytes of memory must be specified

Block of memory is not initialized

Be careful not to access memory outside what you allocated

They return a void pointer

Must be cast before it can be used:

Int \*a = (int \*) malloc( sizeof(int) \* 40 ); //or

Int \*a = (int \*) calloc(40, sizeof(int);

Nov 2

Using structs to create more complex data structures

(linked lists)

arrows (->)

derefs struct pointer and access field to the right

Nov 7

Underlying system calls

-have only seen C library functions so far- have acleaner interface

-several topics to read on:

-one or two languages constructs: enum, union, bitfields, setjmp, longjmp

-process management: system(), popen()

-networking: sockets (covered in Python instead)

Python

-named after Monty Python’s Flying Circus

required reading: python.org, Wikipedia Python article, diveintopython.org, docs.python.org/tut

-school uses 2.7.12, macbook is on 2.7.10…

not compiled but interpreted

program run is python interpreter, takes input source.py

interpreter similar to a.out, so less control over what’s on the cpu

^D closes prompt

importing code/libraries/defaults

import PYTHONFILE

from PYTHONFILE import OBJECT

builtins found by import by default

to add your own append folders to PYTHONPATH env. variable

.py files are found

also sub-folders with (\_init\_.py) (packages)

example

x = os.listdir(“PATH”)

no main program is needed

by default, first unindented piece of code is the first line to run

def defines functions

def fact(n) :

defining main can be helpful to read though, although you will need to call it

Nov 9 (f)

formatting and indenting is very important in python

changing indent style fucks things up- switching between tabs and spaces will do bad things

python types

the dictionary- {idx1:val1…n}

the list- [val1, …valn]

the tuple- like a constant list, (val1, … valn)

the actual types of values are stored and managed internally by python, e.g. float, int, string…

declaring variables: syntax is stupid easy

VARNAME = VALUE

strings- declared in many ways:

s = “hello world”

positional operators: index string[i]

string[i:j] (slice)

length len(string)

formatting: extended printf notation

-“This is %s %.1f” % (“python”, 2.2)

-built in sprint :D

concat with +

python container datatypes

dictionaries

d = {“a”:”b”, “c”:”D”}

>>>d

{“a”:”b”, “c”:”D”}

>>> d[‘a’]

‘b’

>>>d[‘a’] = ‘w’ 🡨 replace

>>>d[“hot”] = “dog” 🡨 auto added

>>>del d[“a”]

>>>d.clear()

-an unordered key-value mapping

-keys must be unique an immutable- means not changeable

lists

-a modifiable, ordered container

-mutable, so cannot be used as a dict key

-defined by [] syntax at creation

-very highly used, and a source of some common errors: copied by reference by default

-mutable types in python are passed by reference in most cases

-this requires care as original data can be destroyed accidentally

-a common solution is to use the built-in “copy” library to achieve pass-by-value

- new\_list = copy.deepcopy(listA)

Nov 11

container practice

lists- see slides, kinda implemented as linked lists

files and try…except

try:

REF = open(“FILENAME”, “TYPE”) #type is r, rb, w, wb, a, ab

except IOError: #if file doesn’t exist, also multiple types of error- IOError, ImportError, etc…

print “oops”

else:

print “if no error”

finally:

print “if error nor not do this”

file stuff:

REF.mode returns mode

REF.name returns filename

REF.tell() returns offset in bytes

args: argv[#] ?

Object definition

Syntax:

class NAME (parentlistbycommas): #parent… like java extends

“””comments”””

def\_\_init\_\_(self,PARAMLISTBYCOMMAS): #constructor. self is like java this

STATEMENTS #other methods go here indented

variables defined within \_\_init\_\_\_

self.NAME = VALUE

by default, variables are public, two underscores before name make private

instantiation

REF = NAME(INIT\_PARAM\_LIST)

Nov 14

Scope in python

-assignment to a variable can either declare a new variable or modify the value of an existing one- depends on scope

LEGB scope rule- local, enclosing, global, python

quirk: python interpreter scans function and determines scope of variable-

def g():

print x

x = 3

return x

x = 2

print g()

will crash because in g(), x is referenced before assignment

referencing global x will work however

Using python and c to write network/web based code

client/server

tcp/ip- streams of packets

discussion of client/server os-level interaction

network sockets:

-basic primitive for network communication

-covered in 310

OS provided abstraction for how data can be exchanged across network

3 classes of internet socket: UDP, TCP, Raw

Sockets:

-addresses (IP addresses)

-port numbers (integers)

commands: ssh, ftp, scp, etc…

-ifconfig: report on network interfaces

ping: send trivial packets to <source>

-iwconfig: manage wireless connections

-iptables, route, etc: turn your pc into a network traffic switch

python socketserver:

4 server classes

tcpserver uses internet tcp protocol

-streams of data between the client and server

-delivery and ordering guarantee

udpserver uses datagrams

-no ordering, no acknowledgement, unreliable

Nov 16

network sockets

xmpp (formerly jabber) is a set of rules for chatting between users mediated by a server

a client connects to a server and identifies itself

a client can send messages to another client

the server handles password verification and manages to message flow

xmpp is a standard and is the basis for google hangouts, facebook chat, msn, etc

xmpp servers usually talk to each other

a public xmpp server can be found at jabber,org

bunch of test clients

less desirable networked systems

bot-net

backdoor

DDOS

HTML: not dynamic

CGI: common gateway interface

allows HTML the ability to embed and run programs

not normal to use other ports than 80 and 443 for HTTP

Nov 18

CGI scripts continued

the stdin of the program is connected to network/server- think redirection or pipes but this is more specialized)

this transmits info from the page including data from forms

stdout of program is returned to network/server, and this must generate the next page

must reside in a web-visible directory

some web servers require that it ends in .cgi. others allow .py, a.out, etc

on mimi, things… follow link in slides to set it up on SOCS

script structure

check form fields- use cgi.FieldStorage class to parse query

takes care of decoding, handles get and post

example dynamic html:

<form name=”adsfasfas” action=”./cgi-bin/asdfasdf” method=”get”>

Nov 21

A long-running dynamic page

-original html file displayed in browser, leads to first call of cgi script

-repeat forever: script outputs html for display… etc

software systems that interface python and c

python efficiency

interpreter adds overhead

types checked at runtime

dynamic realloc

have to be aware how python handles memory in combo with algos

Ctypes:

we’ve seen C code compiled to runnable “programs”- how can a python program use this?

built in python functionality:

ctypes is a foreign function library for python- provides C compatible data types and allows calling functions in DLLs or shared libs- it can be used to wrap these libraries in pure python

from ctypes import cdll

libc = cdll.LoadLibrary(“libc.so.6”)

libc.printf(“Hello, %s\n”, “World!”)

ctypes advantages:

little to no need to midify the C code

ctypes provides many of the necessary interfaces and data tools to make things work on the Python side

disadvantages:

-loose integration between languages

Cython:

is python with C data types

Nov 23

problem with ctypes is…

Cython is another language

written as python- idea is to compile to speed it up

have a setup.py file explaining how to build this:

python setup.py build\_ext –inplace

you end up with a .c file

basically it’s the interpreter, but it autogenerates c code which does the same thing

Nov 25

session maintenance

Correlate requests from same user

-Assign session key on first contact

-incorporate session key in form or in URL

Options:

-in form: use hidden input field

-in url: myhost.com/cgi-bin/myprog.py/thing

Cookies

way for website to store data on client browser

cookies are sent in invisible header that precedes actual html data

website send cookies, you send them back

each cookie has a name, value (key), website, binding

don’t always have to be for security- could just be to maintain sessions

store cookie in browser

can also use https- basically just encrypts all data

how can this be sniffed:

network interactions are broadcast to all comps on a local chunk of network

comp ignores all that are not meant for you, filtered by ip address

we can turn off filtering so process can see all traffic nearby- “promiscuous” mode

CMake version control! finally

source file repository tools

repository is a database that stores versions of a file you’ve been working on- the database permits you to go to a previous version- it also permits you to branch out and experiment with more than one way of dev-ing your file

repositories store any kind of file- .c, a.out, etc

-you can use the database much like the undo command in a text editor but for an entire source file

many popular repositories- RCS and CVS (basic GNU tool), SVN (good), HG and GIT(not root based, good)

good diagram in slides

Dec 5