Warning: These notes are not complete, it is a Skelton that will be modified/add-to in the class. If you want to us them for studying, either attend the class or get the completed notes from someone who did

CSE2031

Introduction

Introduction

- Instructor: Mokhtar Aboelaze
- Room 2026 CSEB
  aboelaze@cse.yorku.ca x40607
- Office hours TR 11:00-12:00 or by appointment

Grading Details

- HW 15%
- Lab 15%
- Midterm 25%
- Final 45%

Introduction

- Course Content
- C
  - Learn how to write test, and debug C programs.
- UNIX (LINUX)
  - Using Unix tools to automate making and testing.
  - Unix shell programming

Text

- The C Programming Language, Kernighan and Ritchie (K+R)
- Practical Programming in the UNIX Environment, edited by W. Sturzlinger
- Class notes (Slides are not complete, some will be filled in during class).
- Man pages

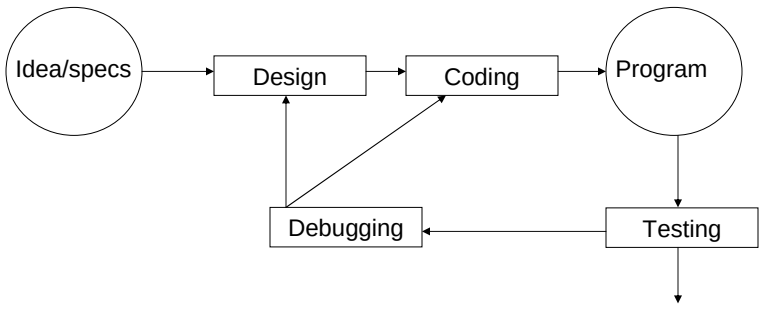
Course Objective

- By the end of the course, you should be able to
  - Write applications (though small) in C
  - Test and debug your code
  - Use UNIX to automate the compilation process
  - Write programs using UNIX shell scripts and awk

WHY C and UNIX

- Wide use, powerful, and fast
- Both started at AT&T Bell Labs
- UNIX was written in assembly, later changed to C
- Many variants of UNIX

Software Development Cycle



Why Testing

- Specifications = LAW, you have to obey it.
- No changes *improvement* unless it is approved
- If in doubt, ask
- First create test cases, test, if error debug repeat
- Testing can show the presence of faults, not their absence -- Dijkstra
- Testing is very costly, in large commercial software 1-3 bugs per 100 line of code.

Why Testing

- 1990 AT&T long distance calls fail for 9 hours
  - Wrong location for C break statement
- 1996 Ariane rocket explodes on launch
  - Overflow converting 64-bit float to 16-bit integer
- 1999 Mars Climate Orbiter crashes on Mars
  - Missing conversion of English units to metric units
- Therac: A radiation therapy machine that delivered massive amount of radiations killing at lease 5 people
  - Among many others, the reuse of software written for a machine with hardware interlock. Therac did not have hardware interlock.

Why Testing

– Jan 13, 2005, LA Times

"A new FBI computer program designed to help agents share information to ward off terrorist attacks may have to be scrapped, forcing a further delay in a four-year, half-billion-dollar overhaul of its antiquated computer system… Sources said about $100 million would be essentially lost if the FBI were to scrap the software…"

Type of Errors

- Errors in program called bugs
- Testing is the process of looking for errors, debugging is found
- Three types of errors
  - Syntax
  - Run-time
  - Logic

Syntax Errors

- Mistakes by violating “grammar” rules
- Diagnosed by C++ compiler
- Must fix before compiler will translate code

Syntax Errors

- #<include stdio.h>
- int main ( );
- {
- printf(“Hello World”);
- /* Next line will output
- a name! /*
- printf(“ Total is %d \n”,total);
- printf(“Final result is \n,result);
- }

```
#include <stdio.h>
int main()
{
printf(“Hello World”);
/*next line will output
A name */
Printf(“Total is %d
\n”,total);
printf(“Final result is
\n”,result););
}
```

Runtime Errors

- Violation of rules during execution of program
- Computer displays message during execution and execution is terminated
- Error message may help locating error

- E.g. X= 5 / 0;

Logical Errors

- Will not be detected by the compiler, may or may not produce an error message (if it results in a runtime error)
- Difficult to find
- Execution is complete but output is incorrect
- Programmer checks for reasonable and correct output

C Syntax

- Java-like (Actually Java has a C-like syntax), some differences
- No //, only /* */ multi line and no nesting
- No garbage collection
- No classes
- No exceptions (try … catch)
- No type strings

First C Program

```
/* Our first program  */
#include <stdio.h>
 void main() {
        printf(“Hello World \n”);
 }
```

Special Characters

\n	New line
\t	Tab
\”	Double quote
\\	The \ character
\0	The null character
\’	Single quote

Data Types

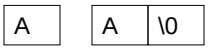
- 4 basic types in C
  - char – Characters
  - int -- Integers
  - float – Single precision floating point numbers
  - double – Double precision floating point numbers

Modifiers

- signed (unsigned) int long int
- long long int
- int may be omitted
- sizeof()

Characters

- One byte
- Included between 2 single quotes
- char x =‘A’
- Character string “This is a string”
- ‘A’ != “A”



- X=‘\012’ newline or 10 decimal

Characters

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Boolean Expressions

- Relational operators
- ==, !=, <, <=, >, >=
- Logical operators
- &&, ||, !

I/O

- Every program has a standard input and output (stdin, stdout and stderr)
- Usually, keyboard and monitor
- Can use > and < for redirection
- printf(“This is a test %d \n”,x)
- scanf(“%x%d”,&x,&y)

%d %s %c %f %lf

integer string character float double precision

I/O

- `int getchar`
  - Returns the next character on standard input or EOF if there are no characters left.
- `int putchar(int c);`
  - Writes the character c on the standard output
- `int printf(char *format,…)`
- `printf(“The result is %f \n”,x);`

C Basics

- Variable name is a combination of letters, numbers, and _ that does not start with a number and is not a keyword
- Abc abc5 aA3_ but not 5sda
- #include <filename.h> replaces the include by the actual file before compilation starts
- #define abc xyz replaces every occurrence of abc by xyz

C Basics

- Expressions
- abc= x+y*z
- J=a%i
- ++x vs. x++
- X += 5;

  // x = x + 5;
- Y /= z;

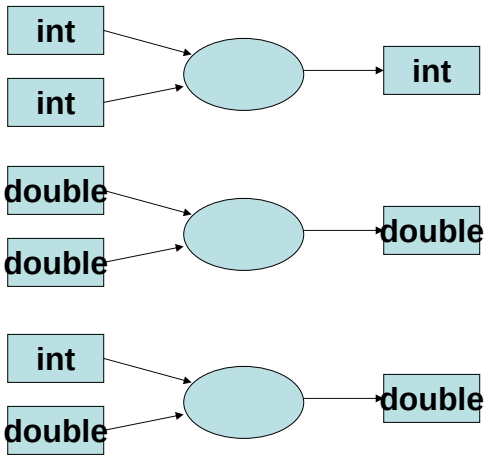
  // Y = Y / z

What is x *= y+1 ?

C Basics

- Decimal numbers 123487
- Octal: starts with 0 0654
- Hexadecimal starts with 0x or 0X ox4Ab2
- 7L for long int =7
- 8U for unsigned
- For floats 24, 23.45, 123.45e-8, 3.4F, 2.15L

Mixed type arithmetic




```
int x=5, y=2, w;
double z, q = 2;

z = x/y;
        // z = 2.0
w = x/y;
        // w = 2
z = x/q;
        // z = 2.5
w = x/q;
        // w = 2
```

Mixed type arithmetic

- 17 / 5
  - 3
- 17.0 / 5
  - 3.4
- 9 / 2 / 3.0 / 4
  - 9 / 2 = 4
  - 4 / 3.0 = 1.333
  - 1.333 / 4 = 0.333

Mixed type arithmetic

- How do you cast variables?

  e.g.

```
int varA = 9, varB = 2;
double  varC;

varC = varA  /  varB;       //  varC is 4.0

varC = varA  /  (double) varB   // varC is 4.5
```

Doesn't change the value of varB, just changes the type to double

Pre- and Post- Operators

- ++ or --
- Place in front, incrementing or decrementing occurs BEFORE value assigned

i = 2 and k = 1

```
k = ++i;        i = i + 1; 3
                k = i;     3

k =--i;         i = i - 1; 1
                k = i;     1
```

- Place in back, occurs AFTER value assigned

i = 2 and k = 1

```
k = i++;        k = i;     2
                i = i + 1; 3

k = i--;        k = i;     2
                i = i - 1; 1
```

Precedence

( )	Parentheses	L to R	1
++, - -	Postincrement	L to R	2
++, - -	Preincrement	R to L	3
+, -	Positive, negative	L to R	3
*, /, %	Multiplication, division	L to R	4
+, -	Addition, subtraction	L to R	5
<=, >=, >, <	Relational operator	L to R	6
==, !=	Relational operator	L to R	7
&&	Logical AND	L to R	8
||	Logical OR	L to R	9
+=, -+, *=, /=, %=	Compound assignment	R to L	10
=	Assignment	R to L	10

Examples

- int a=2, b=3; c=5, d=7, e=11, f=3;
- f +=a/b/c;
- d -=7+c*--d/e;
- d= 2*a%b+c+1;
- a +=b +=c +=1+2;

Bitwise Operators

- Works on the individual bits
- &, |, ^, ~
- short int i=5, j=8;
- k=i&j;
- k=i|j;
- k=~j;

Bit Shifting

- x<<y means shift x to the left y times
- x>>y means shift x to the right y bits
- Shifting 3 many times

0	3
1	6
2	12
3	24
4	48
13	49512
14	32768

Bit Shifting

- What about left shifting
- If unsigned, 0 if signed undefined in C
- It could be logical (0) or arithmetic (sign)
- Unsigned int I =714
- 357 178 89 44 22 11 5 2 1 0
- What if -714
- -357 -178 -89 . . . -3 -2 -1 -1 -1 -1

Examples

Boolean expressions

- False is 0, any thing else is 1

Limits

- The file limits.h provides some constants
- `char- CHAR_BIT, CHAR_MIN, CHAR_MAX, SCHAR_MIN, …`
- int `INT_MIN, INT_MAX, UINT_MAX`
- long `LONG_MIN, …`
- You can find `FLOAT_MIN, DOUBLE_MIN`, … in `<float.h>`

Conditional experssions

- `Test?  exper-true:expe-false`
- `z=(a>b)?  a:b`

Control Flow

- if, while, do while
- The execution of the program depends on some conditions
- Similar to Java

Control Flow

- **if** (`expression`)
- *`statement`*
- **else**
- *`statement`*
- else is optional
- What is statement?

```
; // null statement

x=a+b;

{
    .......
}

{
  ...
    {
     .......
    }
}
```

Control Flow

```
if (expression)
    statement1;
else if (expression)
    statement2;
else if (expression)
    statement3;
else
    statement4;
```

```
if (expression)
    statement1;
  if (expression)
    statement2;
else
    statement4;
```

While

```
while (expression)
    statement
```

```
do
  statement
while(expression)
```

For

- for(i=0, j=3; i<10 && k>2; i++,j--)
- statement
- for(;;)

Break and Continue

- Break – exits the innermost loop
- Continue – skips the current iteration and starts the next one

Switch

- switch(x) {
- case 0 : ……
- break;
- case 1 : …….
- break;
- }

Unique cases, no duplication

Switch (expression) not allowed

Files

- You must open the file before you read or write to it (what about stdin, …).
- The system checks the file, and returns a small non-negative integer known as **file descriptor**, all reads and writes are through this file descriptor.
- 0,1,2 are reserved for stdin, stdout, and stderr.

Files

- FILE *fp1;
- FILE *fopen(char *name, char *mode)

- fp1=fopen(name, mode);
- Name is a character string containing the name of the file, mode is a character string to indicate how the file will be used
- Mode could be “r”, “w”, “a”, “r+b”, ...

Files

- To read or write characters from a file
- `int fgetc(FILE * fp);`
- Returns a byte from a file, or EOF if it encountered the end of file
- `int fputc(int c, FILE *fp);`
- Writes the character c to the file (where to write it?)