week5

Tommy MacWilliam

tmacwilliam@cs50.net

October 10, 2011

Announcements

- quiz0: 10/12 during lecture, not in Sanders
  - one (double-sided) page of notes allowed
- pset3: returned
- pset4: Friday

Today

- Common pset3 mistakes
- Pointers and memory review
- quiz0 review

Hacker Tip of the Week

- looking for a file? you can always `find` it
- `find ~jharvard -name test.c`
  - `~jharvard`: where to start looking
  - `-name` of file

Hacker Tip of the Week

- `find ~/pset3 -name fifteen -type d`
  - `-type` of item is a directory
- `find ~/pset4 -name sudoku -exec /bin/rm '{}' \;`
  - `rm` every file called `sudoku`

pset3

- binary search can be done iteratively or recursive
- algorithm doesn't change, but implementation details can!

pset3

- example time!
  - `search1.c, search2.c, search3.c`

pset3

- magic numbers
  - important, srsly.
- don't forget about `&&` and `||`

```
if (x < 9) {
    if (y < 9) {
    }
}

if (x < 9 && y < 9) {
}
```

Syntax

- &: address of
  - where is the variable located in memory?
- $*$: dereference
  - given a location, go to address to get/set contents
- arrays are just pointers to the first element

Pointers

```
// create a variable
int i;
// create a pointer
int* p;
// set the value of i
i = 5;
// make p point to i
p = &i;
// change the contents of what p points to
*p = 50;
```

Pointers

```
char x = 'a';
char* y = &x;
// address of x (char*)
&x;
// also the address of x (char*)
y;
// address of y (char**)
&y;
// value of x
x;
// also the value of x, because y points there
*y;
// ???
*x;
```

Stack

- local variables are placed on the stack
- each function has its own stack frame, which is inaccessible when the function returns
- grows upward

Heap

- separate from the stack
- values persist even if function returns
    - until you explicitly call `free()`
- allocate space with `malloc`
    - get back the address of the memory you requested
- `free` everything you `malloc`
    - other programs need that memory!

Pointers as Arrays

```
int x = 5;
int* p = malloc(2 * sizeof(int));
*p = 1;
p[1] = 2;
free(p);
```

Reusing Pointers

```
int x = 5;
int* p = malloc(2 * sizeof(int));
free(p);
p = &x;
```

Memory Leaks

- forget to call `free`? memory leak!

```
for (int i = 0; i < 5; i++) {
    int* p = malloc(sizeof(int));
    *p = i;
}
```

Topics

- binary numbers
- compiling code
- `if`, `else`, `while`, `for`
- functions, arguments, return values, recursion
- variable types and scope
- stack and heap
- searching and sorting
- asymptotic notation
- pointers

Binary

42

0	0	1	0	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Compiling Code

- `make sudoku`
  - looks in `Makefile` for `sudoku`
  - runs `gcc` to create binary
  - binary is executable

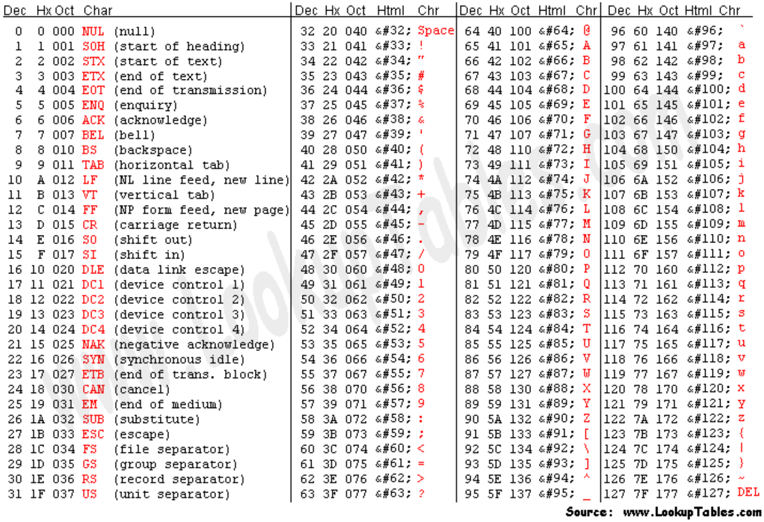
Functions

```
int function(int argument1, float argument2) {
    // do stuff
    int r = 5;
    return r;
}
int x = function(5, 3.14);
```

Variables

- `char`: 1 byte, character
- `int`: 4 bytes, integer
- `float`: 4 bytes, floating-point decimal
- `int*, char*`: 4 bytes, pointer
- `double`: 8 bytes, bigger decimal
- `long long`: 8 bytes, bigger integer

ASCII



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

Source: www.LookupTables.com

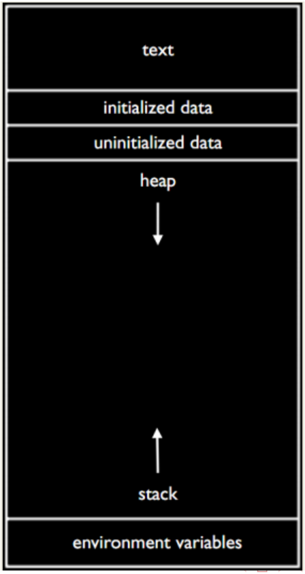
Casting

- `float y = 1.5; int x = (int)y;`
- `char a = 'a'; int b = 'a' + 1;`
- `int c = atoi(“50”);`
- `int d = round(50.5);`

Stack and Heap

- stack: local variables
  - each function gets its own stack frame
  - function returning means stack frame is inaccessible
- heap: `malloc`'d variables
  - persist until you explicitly `free` them

Stack and Heap



Searching

- linear search: look at every single element
  - $O(n)$, $\Omega(1)$
  - does not require sorted list
- binary search: keep looking at middle element
  - $O(\log n)$, $\Omega(1)$
  - requires sorted list

Sorting

- bubble sort: if elements are out of place, swap them
  - $O(n^2)$, $\Omega(n)$
- selection sort: find minimum, put it at the beginning of the list
  - $O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$

Common Running Times

- in ascending order:
  - $O(1)$: constant
  - $O(\log n)$: logarithmic
  - $O(n)$: linear
  - $O(n \log n)$: linearithmic
  - $O(n^c)$: polynomial
  - $O(c^n)$: exponential
  - $O(n!)$: factorial

Questions

IAmA CS50 TF. AMA.

Practice Problems (Doug Lloyd)

- based on the following `giveGrade` function, why is everyone unhappy with their grade?

Practice Problems (Doug Lloyd)

```
char giveGrade(int quizScore) {
  char letterGrade;
  switch(quizScore) {
    case 100: case 90:
      letterGrade = ’A’;
    case 80:
      letterGrade = ’B’;
    case 70:
      letterGrade = ’C’;
    case 60:
      letterGrade = ’D’;
      break;
    default:
      letterGrade = ’F’;
  }
  return letterGrade; }
```

Practice Problems (Doug Lloyd)

- wtf does this do?

```
int secret(int x, int y) {
  if (y == 0)
    return 1;
  else
    return x * secret(x, y-1);
}
```

Practice Problems (Doug Lloyd)

- Write a function `div_by_n()` which takes two arguments, `k` and `n`, and returns `true` if `k` is divisible by `n`, and `false` otherwise.
- Write the few lines of C code that would print out the multiplication table from 1 to 10.

Practice Problems (Doug Lloyd)

- Imagine I execute the following lines of code:

```
string input = GetString();
string input_copy = input;
input_copy[0] = 'X';
```

- Why is it that, if I look at `input[0]`, it is also `'X'`, even though our line of code modified `input_copy`?

Practice Problems

- 2010
  - 2-3, 11, 15, 28, 30, 33
- 2009
  - 22-24, 27
- 2008
  - 3, 6, 13, 15-17

Questions

Last chance.