## ECE 264 Reference Sheet – Spring 2021

command line			
purpose	command	flags	example(s)
view file(s)	<b>ls</b> [-1] [ <i>path</i> ]	$-1 \rightarrow verbose$	ls *.c
change directory	cd directory		cd ps1
make directory	<b>mkdir</b> [-m permissions] directory	$-m \rightarrow set permissions$	mkdir tempdir
remove directory	rmdir directory		rmdir tempdir
delete (remove) files	<b>rm</b> [-r] [-f] <i>path</i>	-r → recursive	rm mytester
copy files	<b>cp</b> [-r] [-f] <i>from</i> to	$-f \rightarrow$ force (remove or	cp -r * backup/
move or rename files	mv from to	overwrite) without asking	mv
view processes	<b>ps</b> [uxw]	uxw→ detailed output	ps auxw
hex dump	<b>xxd</b> [-g # of bytes ]	-g → group by # of bytes	
edit file	<b>vim</b> [-p] <i>path</i>	$-p \rightarrow open files in tabs$	vim -p *.c *.h
compile	<b>gcc</b> [-0 <i>executable</i> ] <i>path</i>	-○ → output executable	gcc -o ps1 ps1.c
get starter files	264get asg	asg is the short name of the	264get hw02
pre-test submission	264test asg		264test hw02
submit	264 submit asg path	<i>path</i> is the file(s) or "*" for all	264submit hw02 *.{c,h}
Submit often and early—ev	en when you are just starting. To restore y	our earlier submission, type <b>264get</b>	help for further instructions.

vim ^ \$ motion h 1 0 W е b to beginning of to end of line to first to beginning to end of this to <u>b</u>eginning of within line ←  $\rightarrow$ line non-blank in line of next word word this or last word motion k j G line# G 응 **m** a-z a-z gg between lines to beginning of to end of file line number to matching mark position go to mark 个  $\mathbf{1}$ file ( { [ < \* motion # **/** pattern pattern n Ν :noh find word, find word, find pattern, \w alphanum or to previous clear search search any char to next . forward backward forward match match highlighting \d number \s whitespace 44 action **\* 7 \* 7** // ~11~11 allall ~~ ~ ~

action	aa	cc	УУ	>>	<<	==	gugu	gugu
current line	<u>d</u> elete line (cut)	<u>c</u> hange line	<u>y</u> ank line (copy)	indent line	dedent line	indent code line	lowercase line	Uppercase line
action	dmotion	<b>c</b> motion	<b>y</b> motion	<b>&gt;</b> <i>motion</i>	< motion	= motion	gumotion	gU motion
by motion	<u>d</u> elete (cut)	<u>c</u> hange	<u>y</u> ank (copy)	indent	dedent	indent code	lowercase	<u>U</u> ppercase
action	i	I	a	A	0	0	P	P
add text	insert before this character	Insert before line beginning	<u>a</u> ppend after this character	<u>A</u> ppend after line end	<u>o</u> pen line below	<u>O</u> pen line above	<u>p</u> ut (paste) text here/below	Put (paste) text before/above
other	v	v	u	^R	•	<b>q</b> <i>a-z</i>	q	@a-z
visual, undo,	visual select	visual select line	undo last action	redo last undone action	repeat last action	record quick macro	stop recording quick macro	play quick macro
commands	: w	:e file	:tabe <i>file</i>	:split	:% <b>s/</b> pattern	/text /gc	<b>:h</b> topic/cmd	÷ط
"ex" mode	<u>w</u> rite (save) file	<u>e</u> dit (open) file	<u>tab</u> : <u>e</u> dit file	split window	replace pattern	n with text	help	quit Vim

 Press
 Esc
 to return to Normal mode.
 Most normal mode commands can be repeated by preceding with a number (e.g., 3dd to delete 3 lines).

 pattern
 may also include:
 \* (×0 or more)
 + (×1 or more)
 + (×0 or 1)
 < (word)
 To rename a variable: :%s/< >/= (zo or 1)

gdb		Controlling quartier	
Start	Automatic display	Controlling execution	View variables and memory
In bash: gdb [tui] file	info display	<u>continue</u>	<pre>print[/[format]] expression</pre>
quit	display expression	finish	• expression : a C expression
set args [arglist]	undisplay [expression#]	jump [file]:function   [file]:line#	<b>x</b> / [# of units] [[unit] ] [format] address
Breakpoints	Explore the stack frame	-next	• # of units : how many units
break [file]:function   [file]:line#	backtrace [full] [n]	return [expr]	• $unit \in b$ (1 byte), h (2 bytes),
clear [file]:function   [file]:line#	down # toward current frame	run [arguments]	W (1 hytes) O (8 hytes)
delete [breakpoint#]	frame [frame#]	<u>set</u> <u>va</u> riable <u>var</u> = <u>expr</u>	
<u>info breakpoints</u>	info args	step	• $format \in d$ (decimal), x (hex),
Watchpoints	info frame	until line#	s (string), f (float), c (character),
watch variable	info locals	Reverse debugging	u (unsigned decimal), O (octal),
awatch variable	list [function   line#[,line#]	record	t (binary), z (zero-padded hex),
rwatch variable	up # toward main(	reverse-next	a (address)
<u>i</u> nfo <u>wat</u> chpoints	whatis variable	reverse-step # and so on.	For more info: <b><u>h</u>elp</b> <u>command</u>
Underlined letters indicate sh	entente la entennost molenno		lanata naramatara that are antional

Underlined letters indicate shortcuts (e.g., n for <u>next</u>, rn for <u>reverse-next</u>, etc.) | Brackets denote parameters that are optional.

course web site: engineering.purdue.edu/ece264/20sp -or- aq.gs/264

## memory

reserved
stack segment
heap segment
BSS segment
data segment
text segment
reserved

Your code, compiled binary text segment
<pre>void oat(char pie) {</pre>
char ham; local variable stack segment
char bun [4]; statically-allocated array stack segment
char* ice = ······ local variable (even an address) · <b>stack</b> segment
"pop"; data segment, read-only
char* yam = ······ local variable (even an address) · <b>stack</b> segment
malloc ( <b>sizeof</b> (*yam)); ·· dynamic allocation block ······· heap segment
<pre>static char egg = 1; static variable, initialized data segment, read-write</pre>
<pre>static char nut; static variable, uninitialized BSS segment</pre>
free(yam);
}

char \_g\_tea; \_\_\_\_\_ global variable, uninitialized \_\_\_\_\_ BSS segment

addresses (pointers)	arrays	strings
int a = 10; // "a gets 10"	2 3 .	char s1[3];
	a1[1] = 8;	<pre>s1[0] = 'H'; // 'H' == 72 s1[1] = 'i'; // 'i' 1== 105 s1[2] = '\0'; // '\0' == 0 char s2[] = {'H', 'i', '\0'};</pre>
	<pre>int a3[2] = {7, 8}; int* a4 = {7, 8};</pre>	<pre>char s3[] = "Hi"; char* s4 = "Hi"; char s5[] = {72, 105, 0}; char s6[] = {0x48, 0x69, 0x00};</pre>
All (a, *b, c, *d) equal 10.	<pre>int* a5 = malloc(     sizeof(*a5) * 2); a5[0] = 7;</pre>	<pre>char s7[] = "\x48\x69"; char* s8 = malloc(sizeof(*s8)*3); strcpy(s8, "Hi"); char* s9 = strdup("Hi"); // non-std</pre>
// arguments (int, int) and returning char."	All (a1a5) contain {7, 8}.	All (s1s9) contain the string "Hi".

## structs

	Basic syntax	Basic syntax + typedef alias	Concise syntax (popular)						
Define struct type	<pre>struct Point {</pre>	<pre>struct _P {</pre>	typedef struct {						
	int x, y;	int x, y;	<pre>int x, y;</pre>						
	};	};	} Point;						
		<pre>typedef struct _P Point;</pre>							
Declare + initialize	<pre>struct Point p = { .x = 10,</pre>	Point p = { .x = 10,	·						
	.y = 20 };	.y = 20 };							
Declare object	struct Point p;		Point p;						
Initialize fields	p.x = 10;  p.y = 20;		·						
Access fields	<pre>int w = p.x; // p.x is the same as (&amp;p) -&gt; x</pre>								
Address (pointer)	<pre>struct Point* a_p = &amp;p Point* p = &amp;p</pre>								
Access via address	<pre>int w = a_p -&gt; x; // a_p -&gt;</pre>	<b>x</b> is the same as (*a_p).x							

linked lists	binary search tree (BST)	merge sort
typedef struct _Node {	typedef struct _BSTNode {	Step 1: Partition the list in half.
int value;	int value;	Step 2: Merge sort each half.
<pre>struct _Node* next;</pre>	<pre>struct _BSTNode* left;</pre>	Step 3: Merge the two sorted halves
} Node;	<pre>struct BSTNode* right;</pre>	into a single sorted list.
	<pre>BSTNode; 4 • • • • • • • • • • • • • • • • • •</pre>	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$

ASC	<b>`II +</b> 2	bla																					
					Charl	D	11.	Charl	D		Charles			Charl		11.	Class			Class		11.	Charles
		Char			Char												Cnar						
	0x20		44	0x2c	,		0x38	8		0x44	D		0x50	P	92	0x5c	\		0x68	h ·		0x74	
	0x21	!	45	0x2d	-		0x39	9		0x45	E		0x51	Q		0x5d			0x69			0x75	
	0x22		46	0x2e	•		ox3a	:		0x46	F		0x52	R		0x5e	^		0x6a			0x76	
	0x23	#	47	0x2f	/		0x3b	;		0x47	G		0x53	S	95	0x5f	_		0x6b	k		0x77	W
	0x24 0x25	\$ %	48	0x30	0		0x3c 0x3d	<		0x48 0x49	H		0x54	T U		0x60 0x61			0x6c 0x6d	 		0x78 0x79	
	0x25 0x26	% &		0x31 0x32	1 2		0x3u 0x3e	= >		<sub>0x</sub> 49 <sub>0x</sub> 4a	- 1		0x55 0x56	V		0x61 0x62	a b		0x60 0x6e			<sub>0x</sub> 79 <sub>0x</sub> 7a	-
	0x20	<u>م</u> ۱	50	0x32	2	62 63	0x3e 0x3f	?		<sub>0x4</sub> a <sub>0x</sub> 4b	K	87	0x50 0x57	W		0x62			0x6e			0x7a 0x7b	
	0x27	1	52	0x33	4		0x31	: @	76	0x40	L		0x57	X		0x03			0x01			0x70	<u> </u>
	0x28	\ \		0x34	5		0x40 0x41	A		0x4C	M		0x58	Y		0x65			0x70	q		0x7C	}
	<sub>0x23</sub>	/ *	55	0x36	6		0x42	B		<sub>0x</sub> 4e	N		<sub>0x</sub> 53	Z		0x66			0x71	ч r		<sub>0x</sub> 7e	-
	0x2b	+		0x37	7		0x42	C	79	<sub>0x</sub> 4f	0		0x5b			0x67	g		0x72			0x7C	
		0660				-		_		-							0	-					
	fine	esso	ر #if	F		#if	def		#el:	50		#nr	agma	ара	ck (	1)		ਸਾਜ	LE			DAI	ास स
	clud		#e]				ndef		#en			#	macro		(strin		-		NE	_		TIN_	
			0.012												1	5 577	_			_		_	
FILE		d str			cha	r*	filo	<u></u>				int		feet	<b>-</b> ( एन	TE *	stre	) )					
гтпр		_			mod		TTTE	name	,			int					E* st		m)				
int		fput	<b>:c</b> (i	nt c	, FI	LE*	str	eam)				int					E* st						
int		_			LE* s							size	e_t							_t s		,	
يديد لا					fmt				£ £ ~ .						_					ream			
int		int			str	eam	, 10	ng o	LISE	36,		SlZe	e_t							: <b>, s</b> i ream	_	t sı	ze,
long					str	ream	)					FILE	*			coun	C <b>,</b> L		50	r cam	/		
int		fget	<b>.c</b> (F	'ILE*	str	ream	)						long <b>ftell</b> (FILE* stream) FILE* <b>stderr</b>										
char	- *	fget	<b>:s</b> (c	har*	buf	, i		, FI	LE*	str				stdo stdi									
		<mark>fget</mark> odes			buf buf		nt n					FILE						a	ddre	ess o	oper	ator	S
		odes		tege		nsta	nt n		twis		eam) Derat	FILE	*	stdi	Ĺn	: 0b <b>10</b> 3	11			ess of		ator	′S & <i>∨</i>
prin	tf co	odes	in	tege	er co	nsta	nt n		t <b>wis</b> bitv	<b>e op</b> wise (	eam) <b>)erat</b> or	FILE Ors Ob10	3* 01	stdi	in 11 ==			ם"ז א"	addre /alue	ess of at <b>a</b> "	V"		&v *a
prin %d %x %c	tf co decin hex char	odes mal acter	in 65 0x 01	tege 41 01	e <b>r co</b> decim hex	nsta	nt n	bit 	t <b>wis</b> bitv bitv	<b>e op</b> wise ( wise a	eam) <b>Oerat</b> or and	FILE Ors 0b10 0b10	01   01 <b>8</b>	stdi	in 11 == 11 ==	00d0	01	ם"ז א"	addre /alue	ess of	V"		&v
prin %d %x %c %p	decin decin hex char addr	odes mal acter	in 65 0x 01 'A'	<b>tege</b> 41 01	er co decim hex octal chara	nsta nal cter	nt n ants	<b>bi</b> t   &	t <b>wis</b> bity bity bity	<b>e op</b> wise (	eam) oerat or and xor	FILE Ors 0b10 0b10 0b10	01   01 & 01 &	stdi 0600	11 == 11 == 11 ==	06 <b>00</b>	01 10	"ז "\ "\	addre value write	ess of at <b>a</b> " <b>v</b> at	<i>v</i> " a"	*a	&v *a
prin %d %x %c %p %s	decin decin hex char addr strin	odes mal acter ess g	in 65 0x 01 'A' '\0	tege 41 01	er co decim hex octal chara null te	nsta nal cter ermir	nt n ants nator	bit   & ^	t <b>wis</b> bity bity bity bity	e op wise ( wise a wise (	eam) oerat or and xor not	FILE OrS 0b10 0b10 0b10 ~ 0b0	01   01 & 01 & 01 ^	0600 0600 0600 0600	11 == 11 == 11 == = 0b1	0b000 0b103	01 10	"; "\ "\ 0	addre /alue write <b>ther</b>	ess of at <b>a</b> "	v" a" erato	*a ors	& <i>v</i> *a != v
prin %d %x %c %p %s	decin decin hex char addr	odes mal acter ess g	in 65 0x 01 'A' '\0	tege 41 01	er co decim hex octal chara	nsta nal cter ermir	nt n ants nator	bit   & ^	twis bity bity bity bity bity	e op wise wise wise wise	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001	0600 0600 0600 111 =	11 == 11 == 11 == = 0b1 2 ==	: 0b000 : 0b10 : 11100 0b000	01 10 000	"a "\ "\ 0	addre /alue write <b>ther</b>	ess of at <b>a</b> " <b>v</b> at <b>ope</b> nary	<b>v</b> " a" erato 3>4 ?	*a ors	&v *a = v == 2
prin %d %x %c %p %s	decin decin hex char addr strin	odes mal acter ess g	in 65 0x 01 'A' '\0	tege 41 01	er co decim hex octal chara null te	nsta nal cter ermir	nt n ants nator	bit   & ~ ~	twis bity bity bity bity bity	e op wise wise wise wise wise	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001	0600 0600 0600 111 =	11 == 11 == 11 == = 0b1 2 ==	: 0b000 : 0b10 : 11100 0b000	01 10 000 00001:	"a "\ "\ 0	addre /alue write ther terr	ess of at <b>a</b> " <b>v</b> at <b>ope</b> nary	<b>v</b> " a" erato 3>4 ?	*a ors 1:2	&v *a = v == 2
prin %d %x %c %p %s %zd	tf co decin hex char addr strin size_	odes mal acter ess g t	in 65 0x 01 'A' '\0 NU	tege 41 01 )' JLL	er co decim hex octal chara null te	nsta nal cter ermir ddres	nt n ants nator ss	bit   & ~ ~ ~	twis bity bity bity bity bity	e op wise wise wise wise wise	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001	0600 0600 0600 111 =	11 == 11 == 11 == = 0b1 2 ==	: 0b000 : 0b10 : 11100 0b000	01 10 000 00001:	"a "\ "\ 0	addre /alue write ther terr	ess of at <b>a</b> " <b>v</b> at <b>ope</b> nary	<b>v</b> " a" erato 3>4 ?	*a ors 1:2	&v *a = v == 2
prin %d %x %c %p %s %zd	tf co decin hex char addr strin size_	odes mal acter ess g t	in 65 0x 01 'A' '\0 NU	tege 41 01 )' JLL	er co decim hex octal chara null te null a	nsta nal cter ermir ddres	nt n ants nator ss	bit   & ~ ~ ~	twis bity bity bity bity bity	e op wise wise wise wise wise	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001	stdi 0600 0600 111 = 11 >>	11 == 11 == 11 == = 0b1 2 ==	: 0b000 : 0b10 : 11100 0b000	01 10 000 00001:	"a "\ "\ 0	addre value write ther terr zeof	ess of at <b>a</b> " <b>v</b> at <b>ope</b> nary	<b>v</b> " a" erato 3>4 ?	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %p %s %zd	tf co decin hex char addr strin size_	odes mal acter ess g t t ence	in 65 0x 01 'A' '\0 NU of a	tege 41 01 )' JLL	er co decim hex octal chara null te null a	nsta nal cter ermir ddres	nt n ants nator ss	bit   & ~ >> << rs a [	twis bity bity bity bity bits	e op wise wise wise wise wise	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001	std: 0600 0600 111 = 11 >> 11 <<	11 == 11 == 11 == 2 == 2 == X	: 0b000 : 0b10 : 11100 0b000	01 10 000 00001:	"a "\ "\ 0	addre value write ther terr zeof	ess of at <b>a</b> " <b>v</b> at <b>ope</b> hary a –	v" a" 3>4 ? size > 2	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %p %s %zd	tf co decin hex char addr strin size_	odes mal acter ess g t t ence	in 65 0x 01 'A' '\0 NU of a	tege 41 01 )' JLL	er co decim hex octal chara null te null a	nsta nal cter ermir ddres	nt n ants nator ss erato	bit   & ^ ~ ~ << rs a [	twis bity bity bity bity bits bits	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001 00111	std: 0600 0600 111 = 11 >> 11 <<	11 == 11 == 11 == 2 == 2 == X	: 0b000 : 0b10 11100 0b000 0b001	01 10 000 00001:	"a "\ "\ 0	addre value write ther terr zeof	ess of at a" v at ope nary a -	v" a" 3>4 ? sized > 2	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %s %zd	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t t ence * a [	in 65 0x 01 'A' '\0 NU <b>of</b> : a 0]	tege 41 01 JLL addr	er co decim hex octal chara null te null a	nsta nal cter ermir ddre: <b>ope</b>	nt n ants nator ss erato	bit   & ~ >> << rs a [	twis bity bity bity bity bits bits	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE 0b10 0b10 0b10 ~ 0b0	01   01 & 01 ^ 00001 00111	std: 0600 0600 111 = 11 >> 11 <<	11 == 11 == 11 == 2 == 2 == X	: 0b000 : 0b10 11100 0b000 0b001	01 10 000 00001:	"a "\ "\ 0	addre value write ther terr zeof	ess of at <b>a</b> " <b>v</b> at <b>ope</b> hary a –	v" a" 3>4 ? sized > 2	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %p %s %zd equ	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t ence * ( a [ of * a	in 65 0x 01 'A' '\0 NU of a a 0] and	tege 41 01 JLL addr	er co decim hex octal chara null to null a ress	nsta nal cter ermir ddres ope	nt n ants hator ss	bit   & ~ ~ << rs a [ { * (a	iwis bity bity bity bity bits bits bits	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 11 <<	11 == 11 == 11 == 2 == 2 == X	: 0b000 : 0b10: 11100 0b000 0b001	01 10 000 000011	"; "\ "\ 1 ?: 5 si	addre value write ther terr zeof	ess of at a" v at r ope nary a - (* a	v" a" 3>4 ? sized > 2	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %s %zd equ effe Addi	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t a [ a [ of * a co a va	in 65 0x: 01 'A' '\0 NU of : a 0 ] and ariab	tege 41 01 JLL addr & or	er co decim hex octal chara null te null a ress null a	nsta nal cter ermir ddre: ope cs * fr	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	iwis bity bity bity bity bits bits +i)	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 0 ( ( ( ( ( ( ( ( ( ( ( ( (	11 == 11 == 11 == 2 == 2 == . X -> varia	: 06000 : 0610: 11100 06000 06001 X	01 10 000 000012 .11100	"a "\ "\ 0 1 ?: 5 si	addre value write ther terr zeof	ess of at a" v at ope nary a – (* a e	v" a" 3>4 ? size > 2 } ) . 2	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %s %zd equ effe Addi	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t t ence a [ a [ of * a co a va	in 65 0x 01 'A' '\0 NU of a 0] and ariab	tege 41 01 JLL addr & or le sul is an	er co decim hex octal chara null to null a ress ress	nsta nal cter ermir ddre: ope cs * fi	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	iwis bity bity bity bity bits bits +i)	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 0 ( ( ( ( ( ( ( ( ( ( ( ( (	11 == 11 == 11 == 2 == 2 == . X -> varia f a is	: 06000 : 0610: 11100 06000 06001 X able a an in	01 10 000 000011 .11100	" " "\ "\ 0 1 ?: 5 si • to it 1	addre value write ther terr zeof	ess of at a" v at ope nary a – (* a (* a e &a is	<pre>v" a" arato 3&gt;4 ? size &gt; 2 } ) . 2 an in</pre>	*a ors 1:2 of(v)	&v *a = v == 2
prin %d %x %c %s %zd equ effe Addi	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t t ence a [ of * a co a va ble: If	in 65 0x 01 'A' '\0 NU of a a 0] and ariab	tege 41 01 JLL addr & or le sul is an is an	er co decim hex octal chara null to null a ress n typ btract int*	nsta nal cter ermir ddre: ope cs * fi	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	iwis bity bity bity bity bits bits +i)	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 0 (0) (1) (1) (1) (1) (1) (1) (1) (1	11 == 11 == 11 == 2 == 2 == . X -> varia f a is f b is	: 06000 : 0610: 11100 06000 06001 X x able a an in an in	01 10 000 000011 .11100 ndds * nt nt	" " "\ "\ 0 1 ?: 5 si • to it 1 1	addre value write ther terr zeof	ess of at a" v at ope nary a - (* a (* a (* a e & a is & b is	<pre>v" a" arate 3&gt;4 ? size &gt; 2 } ) . 2 an in an in</pre>	*a ors 1:2 of(v)	&v *a == v == 4
prin %d %x %c %s %zd equ effe Addi	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t t a [ of * a co a va ole: If	in 65 0x 01 'A' '\0 NU of a 0] and ariab	tege 41 01 JLL addr le sul is an is an is an	er co decim hex octal chara null to null a ress ress	nsta nal cter ermir ddre: ope cs * fi	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	iwis bity bity bity bity bits bits +i)	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 0 (0) (1) (1) (1) (1) (1) (1) (1) (1	11 == 11 == 11 == 2 == 2 == . X -> varia f a is f b is	: 06000 : 0610: 11100 06000 06001 X x able a an in an in	01 10 000 000011 .11100	" " "\ "\ 0 1 ?: 5 si • to it 1 1	addre value write ther terr zeof	ess of at a" v at ope nary a – (* a (* a e &a is	<pre>v" a" arate 3&gt;4 ? size &gt; 2 } ) . 2 an in an in</pre>	*a ors 1:2 of(v)	&v *a == v == 4
prin %d %x %c %s %zd equ effe Addi	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t t a [ of * a co a va ole: If	in 65 0x 01 'A' '\0 NU of a 0] and ariab	tege 41 01 JLL addr le sul is an is an is an	er co decim hex octal chara null to null a ress n typ btract int*	nsta nal cter ermir ddre: ope cs * fi	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	iwis bity bity bity bity bits bits +i)	e op wise o wise o wise o wise o shift o shift o	eam) oerat or and xor not right	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 0 (0) (1) (1) (1) (1) (1) (1) (1) (1	11 == 11 == 11 == 2 == 2 == . X -> varia f a is f b is	: 06000 : 0610: 11100 06000 06001 X x able a an in an in	01 10 000 000011 .11100 ndds * nt nt	" " "\ "\ 0 1 ?: 5 si • to it 1 1	addre value write ther terr zeof	ess of at a" v at ope nary a - (* a (* a (* a e & a is & b is	<pre>v" a" arate 3&gt;4 ? size &gt; 2 } ) . 2 an in an in</pre>	*a ors 1:2 of(v)	&v *a == v == 4
prin %d %x %c %s %zd equ effe Addii	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t t a [ of * a co a va ole: If	in 65 0x 01 'A' '\0 NU of a 0] and ariab	tege 41 01 JLL addr le sul is an is an <b>oper</b>	er co decim hex octal chara null to null a ress ress	nsta nal cter ermir ddre: ope cs * fi * s	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	illians	e op wise o wise o wise o wise o shift o shift o	eam) or and kor not right eft	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00	01   01 & 01 ^ 00001 00111 00111	std: 0600 0600 111 = 11 >> 11 << 0 (0) (1) (1) (1) (1) (1) (1) (1) (1	11 == 11 == 11 == 2 == 2 == . X -> varia f a is f b is	: 06000 : 0610: 11100 06000 06001 X x able a an in an in	01 10 000 000011 .11100 ndds * nt nt	" " "\ "\ 0 1 ?: 5 si • to it 1 1	addre value write ther terr zeof	ess of at a" v at ope nary a - (* a (* a (* a e & a is & b is	<pre>v" a" arate 3&gt;4 ? size &gt; 2 } ) . 2 an in an in</pre>	*a ors 1:2 of(v)	&v *a == v == 4
prin %d %x %c %s %zd equ effe Addii E	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t ence a [ of * a co a va ble: If	in 65 0x 01 'A' '\0 NU of a 0] and and ariab in *n *n *n of c e	tege 41 01 JLL addr is an is an is an is an pper expr	er co decim hex octal chara null te null a ress ress n typ btract int* int ators <i>expr</i> <i>expr</i>	nsta nal cter ermir ddree ope ss * fi * s +++ 	nt n ants hator ss erato	bit   & ^ ~ ~ < * (a ts typ	iwis bity bity bity bity bity bity bity e.	e op wise a wise a wise a shift r shift l	eam) or and kor not right eft	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 Addd E	( &	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	11 == 11 == 11 == 2 == 2 == 2 == varia f a is f b is f c is	: 06000 : 0610: 11100 06000 06001 X able a an in an in an in	01 10 000012 .11100 adds * nt nt* nt*	* to itt 1 1	addre value write ther terr zeof	ess of at a" v at ope nary a - (* a (* a (* a e & a is & b is	<pre>v" a" arate 3&gt;4 ? size &gt; 2 } ) . 2 an in an in</pre>	*a prs 1:2 of(v) 3 4 1:2 (v) 4 5 -= %=	&v *a == v == 4
prin           %d           %x           %c           %p           %s           %zd           equ           effe           Addii           E           ()           []           ->	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t ence a [ of * a co a va ble: If ence expr expr	in 65 0x 01 'A' '\0 NU of a and and and ariab * n * n * n of c * * n	tege 41 01 JLL addr addr is an is an is an is an <i>oper</i> expr expr addr	er co decim hex octal chara null to null a ress ress n typ btract int* int ators expr expr &expr	nsta nal cter ermir ddres ope s * fi * s ++  pr	nt n ants hator ss erato	bit   & ^ ~ ~ <	illians	e op wise a wise a wise a wise a shift r shift r	eam) or and kor not right eft	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 Addc E	( &	std: 0600 0600 111 = 11 >> 11 << 0 (0) (1) (1) (1) (1) (1) (1) (1) (1	11 == 11 == 11 == 2 == 2 == 2 == varia f a is f b is f c is	: 06000 : 0610: 11100 06000 06001 X able a an in an in an in	01 10 000012 .11100 adds * nt nt* nt*	" " "\ "\ 0 1 ?: 5 si • to it 1 1	addre value write ther terr zeof	ess of at $a'''$ v at ope hary a - (*a)	<pre>v" a" arate 3&gt;4? size &gt; 5 } ) . 5 an in an in an in += = /= = ^=</pre>	*a ors 1:2 of(v) 3 4 1:2 (v) 4 (v) 4 (v) 5 (v) (v) 5 (v) (v) (v) (v) (v) (v) (v) (v) (v) (v)	&v *a == v == 4
prin %d %x %c %s %zd equ effe Addii E	tf co decin hex char addr strin size_ ivale	odes mal acter ess g t ence a [ of * a co a va of * a co a va co va co a va co a va co a	in 65 0x 01 'A' '\0 NU of a 0] and ariab ariab **n **n of c **; of c **; **; of c	tege 41 01 JLL addr addr is an is an is an is an <i>oper</i> expr expr addr	er co decim hex octal chara null te null a ress n typ btract int* int ators expr expr &exr f (exp	nsta nal cter ermir ddres ope s * fi * s ++  pr	nt n ants hator ss rato	bit   & ^ ~ ~ <	bity bity bity bity bity bits bits i] + i) e.	e op wise a wise a wise a shift r shift l	eam) or and kor not right eft	FILE OrS 0b10 0b10 0b10 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b00 0b10 0b00	( & ling & k k k k k k k k k k k k k	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	11 == 11 == 11 == 2 == 2 == 2 == varia f a is f b is f c is	<pre>: 0b000 : 0b102 11100 0b000 0b001 X an in an in an in an in</pre>	01 10 000012 .11100 adds * nt nt* nt*	* to it 1 1	addre value write ther terr zeof	ess of at a" v at ope hary a - (*a (*a (*a (*a ka is kc is kc is (*=	<pre>v" a" arate 3&gt;4? size &gt; 5 } ) . 5 an in an in an in += = /= = ^=</pre>	*a ors 1:2 of(v) 3 3 4 1:2 (v) 4 5 5 1:2 (v) 5 5 6 1:2 5 1:2 5 5 1:2 5 5 5 5 5 5 5 5 5 5 5 5 5	&v *a == v == 4

how to write bug-free c			
<ul> <li>DRY – Don't Repeat Yourse</li> </ul>	_	eep and nutrition. • Use assert(	
<ul> <li>Learn to use your tools well</li> </ul>	<ul> <li>Plan before yo</li> </ul>	u begin coding. • Free() where	you malloc(), when possible.
<ul> <li>Fix "broken windows" (e.g.,</li> </ul>	, warnings) • Crash early, e.	g., with assert(). • Design with o	contracts.
and the laboration			
now to debug			
• Test hypotheses systematic			-
<ul> <li>Take notes to stop going in</li> </ul>			tack Overflow, friends, etc.
<ul> <li>Verify your assumptions.</li> </ul>	<ul> <li>Take a nap / wall</li> </ul>	/ break.	andom changes.
memory faults / Valgr	ind arrar massages		
nemory launs / valgi		"Conditional jump or move	"Definitely lost" – leak
	Segmentation fault – crash	depends on uninitialised	Definitely lost – leak
To start Valgrind, run:	Writing at NULL with $*$	value(s)"	Lose address of block
-	<pre>int* a = NULL;</pre>	value(s)	<pre>void foo() {</pre>
valgrind ./myprog	*a = 10;	If with uninitialized condition	<pre>int* a = malloc(); } // !!!</pre>
	Writing at NULL with ->	int a; // garbage!!!	<i>} // ::::</i>
	Node* a = NULL;	<b>if</b> (a == 0) {	"Indirectly lost" – leak
	a -> value = 10;	}	
"Invalid write"			Lose address of address of block
Buffer overflow – heap	Writing at NULL with []	Loop with uninitialized condition	<pre>void foo() {    void** a =</pre>
<pre>int* a = malloc(</pre>	<pre>int* array = NULL; array[0] = 1;</pre>	<pre>int a; // garbage!!! while(a == 0) {</pre>	malloc();
4 * sizeof(*a) );		//	*a = malloc(4);
a[10] = 20; // !!!	Reading from NULL with *	}	} // !!!
Write dangling pointer – heap	int* a = NULL;	Switch with uninitialized condition	
<pre>int* a = malloc();</pre>	<pre>int b = *a;</pre>	int a; // garbage!!!	"Still reachable" – leak
free(a);	Reading from NULL with ->	<pre>switch(a) {</pre>	Address of block still in memory
a[0] = 1;	Node* p = NULL;	//	<pre>int main() {</pre>
	<pre>int b = p -&gt; value;</pre>	}	<pre>static void* a;</pre>
'Invalid read"		Printing unterminated string	a = malloc();
uffer everyand hear	Reading from NULL with []	char s[2];	return EXIT_SUCCESS
Buffer overread - heap int* a = malloc(	<pre>int* array = NULL; int b = array[0];</pre>	s[0] = 'A'; // no '\0'	1
4 * sizeof(*a) );		<pre>printf("%s", s);</pre>	"Invalid free()"
<pre>int b = a[10]; // !!!</pre>			"glibc free"
Read dangling pointer – heap	<pre>int* a = malloc(</pre>	"Use of uninitialized value"	0
int* a = malloc(	10000000000000000);	Passing uninitialized value to fn	Double free
4 * sizeof(*a) );	*a = 1; // a is NULL	int a;	<pre>int* a = malloc(); free(a);</pre>
free(a);	Stack overflow	<pre>printf("%d", a);</pre>	free(a); // !!!
int b = a[0]; // !!!	void foo() {		
	foo(); // !!!	"Syscall param uninitialised	Free something not malloc'd int a = 0;
Not detected by Valgrind	}	byte(s)"	free(&a); // !!!
Buffer overread - stack	Writing to read-only memory	Return uninitialized value from fn	
int a[4];	<pre>char* s = "abc";</pre>	void foo() {	Free wrong part
<pre>int b = a[10]; // !!!</pre>		int a;	<pre>int* a = malloc(); free(a + 3); // !!!</pre>
Buffer overflow – stack		return a;	(u + J), // :::
int a[4];	Calling va_arg too many times while (a == 0) {	}	"silly arg () to malloc()"
a[10] = 1; // !!!	b = va arg();	Write uninitialized value to file	
	} ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	char c;	Negative size to malloc()
		<pre>fwrite(&amp;c, 1, 3, stdout);</pre>	<pre>void* a = malloc(-3); free(a);</pre>
			LTEE(a),

© Copyright 2021 Alexander J. Quinn <aq@purdue.edu> except as noted. This content is protected and may not be shared, uploaded, or distributed. | Versions: This is v1.0.6 (1/19/2021) of this sheet. Content refers to Vim v7.4, GDB v8.3, and Valgrind v3.8. Credits: Bug-avoidance tips inspired by *The Pragmatic Programmer* by Andy Hunt & Dave Thomas | Merge sort image is from *Designing and Building Parallel Programs* © Ian Foster.