ECE 264 Reference Sheet – Fall 2022

command line			
purpose	command	flags	example(s)
view file(s)	ls [-1] [<i>path</i>]	$-1 \rightarrow verbose$	ls *.c
change directory	cd directory		cd ps1
make directory	mkdir [-m permissions] directory	$-m \rightarrow set permissions$	mkdir tempdir
remove directory	rmdir directory		rmdir tempdir
delete (remove) files	rm [-r] [-f] <i>path</i>	-r → recursive	rm mytester
copy files	cp [-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	ps [uxw]	uxw→ detailed output	ps auxw
hex dump	xxd [-g # of bytes]	-g → group by # of bytes	
edit file	vim [-p] <i>path</i>	$-p \rightarrow open files in tabs$	vim -p *.c *.h
compile	gcc [-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	assignment (e.g., "hw01")	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 264get	help for further instructions.

vim								
motion	h	1	0	\$	^	w	е	b
within line	÷	\rightarrow	to beginning of line	to end of line	to first non-blank in line	to beginning of next <u>w</u> ord	to <u>e</u> nd of this word	to <u>b</u> eginning of this or last word
motion	k	j	ga	G	line# G	8	ma-z	I a-z
between lines	1	\checkmark	to beginning of file	to end of file line number		to matching ({ [<	mark position	go to mark
motion	*	#	/ pattern	pa	ttern	n	N	:noh
search	find word, forward	find word, backward	find pattern, forward	. any char	\w alphanum or _ \s whitespace	to next match	to previous match	clear search highlighting
action	dd	cc	УУ	>>	<<	==	gugu	gUgU
current line	delete 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	c <i>motion</i>	y motion	> motion	< 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	Р	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	after <u>o</u> pen line <u>O</u> pen line <u>put (pas end below above here/</u>		put (paste) text here/below	Put (paste) text before/above
other	v	v	u	^R	•	q <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 file	:split	:% s/ pattern	/text /gc	:h topic/cmd	:q
"ex" mode	write (save) file	edit (open) file	tab: edit 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) + (×1 or more) + (×0 or 1) / (word) | To rename a variable: :*s//

gab			
Start	Automatic display	Controlling execution	View variables and memory
In bash: gdb [tui] file	<u>i</u> nfo <u>di</u> splay	<u>c</u> ontinue	<pre>print[/format] expression</pre>
quit	display expression	finish	• expression : a C expression
set args [arglist]	undisplay [expression#]	jump [file]:function [file]:line#	• / [# of units] [[unit]] [format] address
Breakpoints	Explore the stack frame	-next	• # of units] [units] [units] [units] [units]
break [file]:function [file]:line#	<u>b</u>ack<u>t</u>race [full] [n]	run [arguments]	• $unit \in b$ (1 byte), h (2 bytes),
delete [breakpoint#]	down # toward current frame	set variable var = expr	w (4 bytes), g (8 bytes)
info breakpoints	info args	step	• $format$ \in d (decimal), x (hex),
Watchpoints	info frame	until line#	S (string), ± (float), C (character),
watch variable	<u>i</u> nfo <u>lo</u> cals	Reverse debugging	u (unsigned decimal), O (octal),
awatch variable	list [function line#[,line#]	record	t (binary), Z (zero-padded hex),
<u>rw</u> atch <u>variable</u>	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: <u>help</u> command

course web site: engineering.purdue.edu/ece264/22au -or- aq.gs/264

memory

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

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

char _g_tea; _____ global variable, uninitialized _____ BSS segment

addresses (pointers)	arrays	strings
int a = 10; // "a gets 10"	int a1[2];	char s1[3];
<pre>int* b; // "b is an address of an int" b = &a // "b gets the address of a" int c = *b; // "c gets the value at b"</pre>	a1[0] = 7; a1[1] = 8; int a2[] = {7, 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* d = malloc(sizeof(*d)); // "d gets the address of a new allocation block // "d gets the address of a new allocation block</pre>	<pre>int a3[2] = {7, 8}; int* a4 = {7, 8};</pre>	<pre>char s3[] = "Hi"; char* s4 = "Hi"; char s5[] = {72, 105, 0};</pre>
<pre>// sufficient for 1 int" *d = 10;</pre>	<pre>int* a5 = malloc(sizeof(*a5) * 2);</pre>	<pre>char s6[] = {0x48, 0x69, 0x00}; char s7[] = "\x48\x69"; char* s8 = malloc(sizeof(*s8)*3);</pre>
char (*a_f) (int, int) = f; // "a_f is the address of function f() taking 2 // arguments (int, int) and returning char."	a5[0] = 7; a5[1] = 8; All (a1a5) contain {7, 8}.	<pre>strcpy(s8, "Hi"); char* s9 = strdup("Hi"); // non-std All (s1s9) contain the string "Hi".</pre>

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	struct Point p = { .x = 10,	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	int w = p.x; // p.x is the same as (&p) -> x							
Address (pointer)	<pre>struct Point* a_p = &p</pre>		Point* p = &p					
Access via address	<pre>int w = a_p -> x; // a_p -></pre>	x is the same as (*a_p).x						

linked lists	binary search tree (BST)	merge sort
<pre>typedef struct _Node {</pre>	typedef struct _BSTNode {	Step 1: Partition the list in half.
<pre>int value;</pre>	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;	struct BSTNode* right;	into a single sorted list.
$2 \bullet \bullet 6 \bullet \bullet 4 \bullet \bullet$	BSTNode; 4	$\begin{array}{c} 6 \\ \hline \\ 2 \\ \hline 2 \\ 2 \\$
		$\begin{array}{c c} 9 \\ \hline 9 \\ \hline 5 \\ \hline \end{array} \begin{array}{c} 9 \\ \hline 9 $

AS(CII ta	ble																					
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	0x20	11	44	0x2c	,	56	0x 38	8	68	0x44	D	80	0x50	Р	92	0x5c	\	104	0x68	h	116	0x74	t
33	0x21	!	45	0x2d	-	57	0x39	9	69	0x45	E	81	0x51	Q	93	0x5d]	105	0x69	i	117	0x75	u
34	0x22	"	46	_{0x} 2e		58	_{0x} 3a	:	70	0x46	F	82	0x52	R	94	_{0x} 5e	۸	106	_{0x} 6a	j	118	0x76	v
35	0x23	#	47	0x2f	/	59	0x3b	;	71	0x47	G	83	0x53	S	95	0x5f		107	0x6b	k	119	0x77	w
36	0x24	\$	48	0x30	0	60	0x3c	<	72	0x48	Н	84	0x54	Т	96	0x60	`	108	0x6c	Ι	120	0x 78	х
37	0x25	%	49	0x31	1	61	0x3d	=	73	0x49	I	85	0x55	U	97	0x61	а	109	0x6d	m	121	0x79	у
38	0x26	&	50	0x32	2	62	0x3e	>	74	_{0x} 4a	J	86	0x56	V	98	0x62	b	110	0x6e	n	122	0x7a	Z
39	0x27	' 	51	0x33	3	63	0x3f	?	75	0x4b	K	87	0x57	W	99	0x63	C	111	0x6f	0	123	0x7b	{
40	0x28	(52	0x34	4	64	0x40	@	76	0x4c	L	88	0x58	X	100	0x64	d	112	0x70	р	124	0x7c	
41	0x29)	53	0x35	5	65	0x41	A	//	0x4d		89	0x59	Y	101	0x65	e	113	0x/1	q	125	0x/d	}
42	0x2a	т 	54	0x36	6 7	60	0x42	<u>с</u>	/8 70	0x4e	N 0	90	0x5a	ک ۲	102	0x66	T a	114	0x/2	r	126	0x/e	
45	UXZD	Ŧ	55	0x57	/	07	0x45	C	79	0x41	0	91	UXSD	<u> </u>	105	UX O 7	g	115	UX75	5	127	UX71	DEL
pre #do	oroc finc	esso	or #if	-		#;f	dof		#~1	60		#nr	2077		ak (1)		ਯ ਜ	тъ			יעם	יםי
#ue #in	clud	le	#=1 #e]	- Lif		#if:	ndef		#en	se d		#PL	macro	, pa	(strin	∟) aifv)	-	r _ LI	NE	_		TIN	1E 1E
filos	and	d etr	oam	ne –											• •					_			
FIL		fope		onst	cha	ar*	file	name				int		feot	E(FI	LE *	stre	am)					
		cons	t cl	har*	mod	e)			,			int		fer	ror (FILE	I* st	rea	m)				
int		fput	c(i	nt c	, Fl	LE*	str	eam)				int		fclo	ose (FILE	l* st	rea	m)				
int		fpri	ntf	(FII	E* :	stre	am,					size	e_t	frea	ad (v	oid*	des	st,	size	_t s	ize,	,	
int		cons fsoc	CI	nar^ 'TT.E.*	IMU	, .	· ·)	ng o	ffq	₋ +			~ +	size	e_t (coun	t, F	ILE'	st:	ream)	- ai	F 0
LIIC		int	whe	nce)	SUI	cam	, 10	iig o	113	ει,		SIZE	ະ_ເ	size	et (coun	t. F	ILE,	sic st:	, sı ream	.ze_')	LSI	ze,
long	J	ftel	ll (F	'ILE*	str	ream)					FILE	*	stde	err	00011	0, 1		001	- o ani	/		
int		fget	.c (F	'ILE*	stı	ream)					FILE	*	stdo	out								
char* fgets (char* buf, int n, FILE* stream)FILE* stdin																							
		rget	:s (c	har*	buf	:, i	nt n	, FI	LE*	stre	eam)	FILE	· *	std	in			_					
prin	tf co	odes	cs (c in	^{har*} tege	buf r co	i, i nsta	nt n ants	, FI bit	∷LE* twis	stro e op	eam) Derat	FILE Ors	2*	std	in			a	ddre	ess o	oper	ator	S
prin %d	tf co decir	ndes mal	s (c in 65	har* tege	buf r co decim	≣, i nsta nal	nt n ants	, _{FI} bit	twis bit	stro e op wise (eam) erat or	FILE Ors 0b10	E* 01	std: 0b00	in 11 ==	0b 10 2	11	a "a	ddre addre	ess of	oper v"	ator	s &v
prin %d %x	tf co decir hex	nal	s (c in 65 0x	har* tege (41	buf r co decim nex	nsta nal	nt n ants	, FI bit &	t wis bit bit	stro e op wise o wise a	eam) eraí or and	FILE O rs 0b10 0b10	2* 01 01 8	std: 0600	in 11 == 11 ==	0b 10 :	11 01	a "a "\	ddre addre value	ess of at a"	oper v"	ator	s & <i>v</i> *a
prin %d %x %c	decir decir hex char	nal	65 0x4 01	har* tege (41 01 (buf r co decim nex octal	nsta nal	nt n ants	, ⊮I & ∧	twis bit bit bit	stro e op wise o wise a wise a	eam) oerat or and kor	FILE OrS 0b10 0b10 0b10	01 01 01 8 01 ^	std: 0600 0600	in 11 == 11 == 11 ==	0b10 0b000	11 01 10	a "a "\	ddre addre value write	ess of at a" v at a	oper v" a"	ator *a	S &v *a = v
prin %d %x %c %p	decir decir hex chara addr	nal acter ess	65 0x 01 'A'	har* tege (41 01 (buf r co decim nex octal chara	f, i nsta nal cter	nt n ants	, FI & ^ ~	LE* twis bit bit bit bit	stre e op wise o wise a wise o wise o	eam) oerat or and kor not	FILE Ob10 0b10 0b10 0b10 ~ 0b	2* 01 01 & 01 ^ 00001	ob00 0b00 0b00	in 11 == 11 == 11 == = 061	0b10: 0b000 0b10: 11100	11 01 10 000	a "c "\ "\	ddre addre value write ther	ess of at a" v at a ope	oper V" a" erato	ator *a	s &v *a = v
prin %d %x %c %p %s	tf co decir hex chara addr strin	nal acter ess	65 0x 01 'A'	har* tege 41 01 (()	buf r co decim nex octal chara null to	E, i nsta nal cter ermin	nt n ants nator	, FI & ~ ~ >>	LE* twis bit bit bit bit	stre e op wise o wise o wise o wise o shift r	eam) or and kor not right	FILE Ob10 Ob10 Ob10 Ob10 ~ Ob0 Ob00	01 01 8 01 8 001 ^ 00001	std: 0600 0600 0600 1111 = 112>	11 == 11 == 11 == = 0b1 2 == 0	: 0b10: : 0b00(: 0b10: 1110(0b000	11 01 10 000	a "a "\ "\ 0 ?:	ddre addre value write ther	ess of ss of at a" v at a ope	oper v" a" rato 3>4 ?	ator *a ors 1:2	S & <i>v</i> *a = <i>v</i> == 2
prin %d %x %c %p %s %zd	tf co decir hex chara addr strin size_	odes mal acter ess g	65 0x 01 'A' '\0	har* tege (41 01 (()' JLL	buf r co decim nex octal chara null to null a	nstanal nal cter ermin ddre	nt n ants nator ss	, FI bit 8 ^ ~ >> <<	LE* bit bit bit bit bit bit	stre e op wise o wise o wise o wise o shift r	eam) or and kor not right eft	FILE 0b10 0b10 0b10 ~ 0b10 0b00 0b00	01 01 & 01 ^ 00001 00011	std: 0b00 0b00 0b00 1111 = 11 >>	11 == 11 == 11 == 2 == 0 2 == 0	: 0b10: : 0b00(: 0b10: 1110(0b000 0b001	11 01 10 000 00011 11100	a "a "\ "\ 0 ?:) si	ddre addre value write ther terr zeof	ess of ss of at a" v at a ope ary	oper v" a" srato 3>4 ?	*a *a ors 1:2 of(<i>v</i>)	S & <i>V</i> * a = <i>V</i> == 2 == 4
prin %d %x %c %p %s %zd	tf cc decir hex chara addr strin size_	nal acter ess g t	65 0x 01 'A' '\0 NU	har* tege (41 01 () JLL	buf r co decim nex octal chara null to null a	nsta nal cter ermin ddre	nt n ants nator ss	, FI bit & ~ ~ >> <<	LE* bit bit bit bit bit bit	stre e op wise o wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE 0b10 0b10 0b10 ~ 0b0 0b00 0b00	01 01 8 01 ^ 00001 00011:	std: 0600 0600 0600 1111 = 11 >> 11 <<	11 == 11 == 11 == 2 == 2 ==	: 0b10: : 0b00(: 0b10: 1110(0b000 0b001	11 01 000 00011 11100	a "{ "\ "\ 0 ?: si	ddre addre value write ther terr zeof	ess of at a" v at a ope	oper v" a" srato 3>4 ? size	*a ors 1:2 of(v)	S & <i>v</i> * a = <i>v</i> == 2 == 4
prin %d %x %c %p %s %zd	tf co decir hex chara addr strin size_	odes mal acter ess g t	(c)	har* tege 41 01 (0)' JLL	buf r co decim nex octal chara null to null a	nstanal cter ermin ddre	nt n ants nator ss	, FI & ^ ~ >> << rs	twis bit bit bit bit bit bit	stro wise o wise o wise o wise o wise o shift o shift o	eam) or and kor not right eft	FILE 0b10 0b10 0b10 ~ 0b0 0b00	01 01 & 01 ^ 00001 00011:	std: 0600 0600 0600 1111 = 11 >>	in 11 == 11 == = 0b1 2 == 0	: 0b10: : 0b000 : 0b10: 11100 0b000 0b001	11 01 10 000 00011 11100	a "a "\ "\ 0 ?:	ddre addre value write ther terr zeof	ess of at a" v at a ope	oper v" a" srato 3>4 ? sized	*ator *a ors 1:2 of(v)	S & <i>v</i> * a = <i>v</i> == 2 == 4
prin %d %x %c %p %s %zd	tf co decir hex chara addr strin size_ ivale	ndes mal acter ess g t ence	<pre>65 65 0x 01 'A' '\0 NU of a </pre>	har* tege 41 01 d 01 d 01 d 01 d 01 d 01 d 01 d 01 d	buf decim nex octal chara null to null a	ermin ddre	nt n ants nator ss	, FI & ^ ~ >> << rs a [LE* bit bit bit bit bit	stro e op wise o wise o wise o wise o shift n shift l	eam) or and kor not right eft	FILE 0b10 0b10 0b10 ~ 0b0 0b00	01 01 & 01 ^ 00001 0011:	std: 0600 0600 0600 111 = 11 >> 11 <<	11 == 11 == 11 == 2 == 0 2 == 0	: 0b10: : 0b00(: 0b10: 11100 0b000 0b001	11 01 000 00011 11100	a "t "\ "\ 0 si	ddre addre value write ther terr zeof	ess of at a" v at a ope ary	oper V" a" sized > 2	*a *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> == <i>v</i> == 2 == 4
prin %d %x %c %p %s %zd	tf co decir hex chara addr strin size_	odes mal acter ess g t ence	<pre>c (c in 65 0x 01 'A' '\0 NU of a a</pre>	har* tege 41 01 ()' JLL	buf r CO decim nex octal chara null to null a	f, i nsta nal cter ermin ddre ope	nt n ants nator ss erato	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	LE* bit bit bit bit bit bit	stro e op wise o wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 ~ Ob0 Ob00 Ob00	01 01 & 01 ^ 00001 0011:	std: 0600 0600 0600 111 = 11 >> 11 << 0	in 11 == 11 == 2 == 2 == X	: 0b10: : 0b00(: 0b10: 11100 0b000	11 01 10 000 00011 11100	a "'a "\ "\ ?: o si	ddre addre value write ther tern zeof	ess of ss of at a" v at o ope ary a –	oper v" a" srato 3>4 ? sized > 2	*ator *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> = <i>v</i> == 2 == 4
prin %d %x %c %p %s %zd	tf co decir hex chara addr strin size_	nal acter ess g t ence * a [<pre>cs (cc s) (</pre>	har* tege 41 01 d 1 JLL addr	buf r co decim nex octal chara null to null a	E, i nsta nal cter ermin ddre ope	nt n ants nator ss	, FI & ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<pre>LE* twis bit bit bit bit bit bit bit bit bit +i)</pre>	stro e op wise o wise o wise o wise o shift r shift l	eam) erat or and kor not right eft	FILE Ob10 Ob10 Ob10 ~ Ob1 Ob00 Ob00	01 01 & 01 ^ 00001 00011: 00011:	std: 0600 0600 111 = 11 >> 11 << (20)	in 11 == 11 == 2 == 0 2 == 0 X ->	: 0b10: : 0b00(: 0b10: 11100 0b000 0b001	11 01 000 00011 11100	a "t "\ "\ 0 ?:) si	ddre addre value write ther tern zeof	ess of ss of at a" v at a ope ary ary	oper v" a" sized > 2) . 2	*ator *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> == <i>v</i> == 4
prin %d %x %c %p %s %zd equ	tf co decir hex chara addr strin size_ ivale	odes mal acter ess g t ence * 2 a [<pre>cs (cc s) (</pre>	har* tege 41 01 0 1 JLL addr	buf r co decim nex octal chara null to null a ess	nstanal nal cter ermin ddre ope	nt n ants nator ss	, FI & ^ ~ ~ << rs a [{ }	<pre>LE* twiss bit bit bit bit bit bit bit bit bit bit</pre>	stro e op wise o wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 ~ Ob0 Ob00 Ob00	01 01 8 01 ^ 00001 00011: 00011:	std: 0600 0600 111 = 11 >> 11 << 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 0 12 12 12 12 12 12 12 12 12 12	in 11 == 11 == 2 == X ->	: 0b10; : 0b00 : 0b10; 11100 0b000 0b001	11 01 10 000 00011 11100	a "; "\ "\ 0 . ?:	ddre addre value write ther terr zeof	ess of ss of at a" v at a ope ary a - (*a	> per v" a" srato 3>4 ? sized > 2) . 2	*ator *a ors 1:2 of(v)	S & <i>v</i> * a = <i>v</i> == 2 == 4
prin %d %x %c %p %s %zd equ	tf co decir hex chara addr strin size_ ivale	nal acter ess g t a [of * a co a vio	<pre>cs (cc s, cc s, cc</pre>	har* tege 41 01 0 JLL 1 addr & or le sub	buf r co decim nex octal chara null to null a ess	r, i nsta nal cter ermin ddre ope ts * f	nt n ants hator ss erato	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	LE* twis bit bit bit bit bit bit bit bit	stro e op wise o wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00 Ob00 Ob00 Ob00	01 01 & 01 ^ 00001 00011 00011	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	in 11 == 11 == 2 == 0 2 == 0 X -> varia	: 0b10: : 0b00 : 0b10: 11100 0b000 0b001 X	11 01 000 00011 11100	a "c "\ "\ 0 ?:) si	ddre addre value write ther tern zeof	ess of ss of at a" v at a ope ary ary (* a	> per v" a" erato 3>4 ? sized > 2) . 2	*ator *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> == <i>v</i> == 4
prin %d %x %c %p %s %zd equ effe Addi	tf co decir hex chara addr strin size_ ivale	acter ess g t a [of * a co a value:	<pre>cs (cc since show in the second show in the se</pre>	har* tege 41 01 0 JLL 1 addr & or le sul is an	buf r CO decim nex octal chara null to null a ess r typ otract int*	<pre>f, i nsta nal cter ermin ddre ope ts * f c*</pre>	nt n ants hator ss erato	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<pre>LE* twiss bit bit bit bit bit bit bit bit bit bit</pre>	stro e op wise o wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00 Ob00 Ob00 Ob00	01 01 & 01 ^ 00001 0011: 0011: (&	std: 0600 0600 0600 111 = 11 >> 11 << 0 20) 2 to a ple: 1	in 11 == 11 == 2 == 0 2 == 0 . X -> varia f a is	: 0b10: : 0b00 : 0b10: 11100 0b000 0b001 X able a an in	11 01 10 000 111100 dds *	a "; "\ "\ 0 ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	ddre addre value write ther terr zeof s type then	ess of at a" v at a ope ary a - (*a (*a &a is	> per v" a" sized > 2) . 2 an ir	*ator *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> = <i>v</i> == 2 == 4
prin %d %x %c %p %s %zd equ effe Addi	tf cc decir hex chara addr strin size_ ivale	odes mal acter ess g t a [co a va ole: If	<pre>s (cc s; (c</pre>	har* tege 41 01 0 1 JLL 1 addr le sul is an is an	buf r CO decim nex octal chara null to null a ess typ otract int*	<pre>E, i nsta nal cter ermin ddre ope ts * fi c*</pre>	nt n ants hator ss rato	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<pre>LE* twis bit bit bit bit bit bit bit bit bit bit</pre>	stro e op wise o wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob00 Ob10 Ob00	01 01 8 01 ^ 00001 00011: 00011: (& Ling 8	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	in 11 == 11 == 2 == 2 == x yaria f a is f b is	: 0b10: : 0b00 : 0b10: 11100 0b000 0b001 X able a an in an in	11 01 10 000 11100 dds * nt nt	a "c "\ "\ 0 : : : : : : : : : : : : : : : : : :	ddre addre value write ther terr zeof	ess of ss of at a" v at a ope ary a - (*a (*a & a is & b is	oper v" a" srato 3>4 ? sized > 2) . 2 an ir an ir	*ator *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> == <i>v</i> == 4
prin %d %x %c %p %s %zd equ effe Addi	tf co decir hex chara addr strin size_ ivale	acter ess g t a [of * a of * a	<pre>s (c since state st</pre>	har* tege 41 01 0 1 JLL 1 addr addr le sul is an is an is an	buf r CO decim nex octal chara null to null a ess typ otract int* int*	<pre>E, i nsta nal cter ermin ddre ope ts * f c*</pre>	nt n ants hator ss erato	, FI & ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<pre>LE* twis bit bit bit bit bit bit bit bit bit bit</pre>	stro e op wise o wise o wise o wise o shift n shift n	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00 Ob00 Ob00 Ob00 Ob00	01 01 & 01 ^ 00001 0011: 0011: (&	std: 0600 0600 111 = 11 >> 11 << 0 20) 2 to a ple: 	11 == 11 == 11 == 2 == 0 2 == 0 2 == 0 2 == 0 varia f a is f b is f b is f c is	: 0b10: : 0b00 : 0b10: 11100 0b000 0b001 X able a an in an in an in	11 01 10 000 11100 dds * ht ht*	a "{ "\ "\ 0 ?:) si to it 1 1	ddre value write ther terr zeof s type then then	ass of at a" vata ope ary ary (*a (*a (*a s &a is &b is &c is	<pre>> per v" a" rato 3>4 ? sized > 2) . 2 an ir an ir an ir</pre>	*ator *a ors 1:2 of(v)	S & <i>v</i> * <i>a</i> = <i>v</i> == 2 == 4
prin %d %x %c %s %zd equ effe Addi	tf co decir hex chara addr strin size_ ivale cts o ng * t xamp	acter ess g t a [of * a co a va ence	<pre>s (c since si</pre>	har* tege 41 01 0 1 JLL 1 addr le sul is an is an is an oper	buf r CO decim nex octal chara null to null a ess to typ otract int* int ator	nstanal nal cter ermin ddre ope ts * f	nt n ants hator ss erato	, FI & ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	LE* twis bit bit bit bit bit bit bit bit	stro e op wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE 0ь10 0ь10 0ь10 ~ оь0 0ь00 0ь00	01 01 & 01 ^ 00001 0011: 0011: (&	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	11 == 11 == 11 == 2 == 2 == X yaria f a is f b is f c is	: 0510: : 0500(: 0510: 11100 05000 05001 X able a an in an in an in	11 01 10 000 11100 dds * ht ht*	a "c "\ "\ 0 ?:) si to it 1 1	ddre addre value write ther terr zeof s type then then then	ass of at a" vat ope ary ary (*a (*a (*a sa is sb is sc is	oper v" a" sized > 2) . 2 an ir an ir an ir	*ator *a ors 1:2 of(v)	s &v *a == v == 4
prin %d %x %c %p %s %zd equ effe Addi	tf co decir hex chara addr strin size_ ivale cts o ng * t Examp	acter ess g t a [of * a of * a of * a of * a of * a	<pre>s (c s (c s s</pre>	har* tege 41 01 0 JLL 1 addr addr le sul is an is an is an <i>oper</i> <i>expr</i>	buf r CO decim nex octal chara null to null a ess otract int* int ators expr	<pre>E, i nsta nal cter ermin ddre ope ts * fi s * s</pre>	nt n ants nator ss erato	, FI 	LE* twis bit bit bit bit bit bit bit bit	stro e op wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00 Ob00 Ob00 Ob00 Ob00	01 01 & 01 ^ 00001 0011: 0011: (& ling &	std: 0600 0600 111 = 11 >> 11 << (0) (0) (0) (0) (0) (1	11 == 11 == 11 == 2 == 2 == 2 == x yaria f a is f b is f c is	: 0510: : 0500 : 0510: 11100 05000 05001 X x able a an in an in an in	11 01 10 000 11100 dds * ht ht*	a "; "\ ?:) si to it 1 1	ddre addre value write ther zeof s type then then then	ess of at a" v at a ope ary ary (* a (* a & a is & b is & c is	<pre>> per v" a" rato 3>4 ? sized > 2) . 2 an ir an ir an ir +=</pre>	*a ors 1:2 of(v) C C A A A A A A A A A A A A A	S & <i>v</i> * <i>a</i> = <i>v</i> = <i>2</i> = <i>4</i>
prin %d %x %c %p %s %zd equ effe Addi	tf cc decir hex chara addr strin size_ ivale cts c ng * t Examp	acter ess g t acter ess g t a [co a v co a	<pre>s (c s (c s s</pre>	har* tege 41 01 0 JLL 1 addr addr le sul is an is an is an is an expr expr	buf r CO decim nex octal chara null to null a ess otract int* int* int expr expr	<pre>E, i nsta nal cter ermin ddre ope ts * f ts * f s s ++ ++ </pre>	nt n ants hator ss rom it . then	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<pre>LE* twis bit bit bit bit bit bit bit bit bit bit</pre>	stro e op wise o wise o wise o shift r shift l	eam) or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00 Ob00 Ob00 Ob00 Db00 Ob00	01 01 8 01 ^ 00001 00011 00011 (& Ling 8	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	<pre>in 11 == 11 == 11 == 2 == 2 == . X . </pre>	x an in an in an in an in	11 01 10 000 11100 dds * ht ht* ht*	a "c "\ "\ "\ ?:) si to it 1 1	ddre addre value write ther terr zeof	ess of ss of at a" v at a ope ary a - (* a (* a & a is & b is & c is	<pre>> per v" a" rate 3>4 ? sized > 2 }) . 2 an ir an ir an ir += = /=</pre>	<pre>*ator *a ors 1:2 of(v) 3 att att att att att att att att att a</pre>	s &v *a == v == 4
prin %d %x %c %p %s %zd equ effe Addi	tf cc decir hex chara addr strin size_ ivale cts c ng * t Examp	acter ess g t a [a [a [a [cof * a co a v cof * a co a v cof * a co a v co a	<pre>s (c since set of set of</pre>	har* tege 41 01 0 JLL 1 addr addr is an is an is an is an is an <i>oper</i> <i>expr</i> <i>expr</i> <i>expr</i> <i>addr</i>	buf r CO decim nex octal chara null to null a ess otract int* int int* int expr expr &exp	<pre>E, i nsta nal cter ermin ddre ope ts * f f * S *++ pr</pre>	nt n ants hator ss rato	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	LE* twis bit bit bit bit bit bit bit bit	stro e op wise o wise o wise o wise o shift r shift l	eam) erat or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00	5 * 01 01 & 001 ^ 00001 00011 0011 (& ling & Examp	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	<pre>in 11 == 11 == 11 == 2 == 2 == varia f a is f b is f c is </pre>	 0b10: 0b10: 0b10: 0b000 0b000 0b001 X an in an in an in 	11 10 000 00011 11100 dds * ht ht* ht*	a "c "\ "\ ?:) si to it 1 1 1	ddre addre value write ther terr zeof s type then then then	ess of ss of at a" v at a ope ary a - (*a (*a &a is &b is &c is &c is	<pre>> per v" a" rato 3>4 ? sized > 2) . 2 an ir an ir an ir an ir = /= = /= = ^=</pre>	<pre>*ator *a ors 1:2 of(v) 4 of(v) 4 of(v) 4 of(v) 5 c c c c c c c c c c c c c c c c c c</pre>	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>
prin %d %x %c %p %s %zd equ effe Addi	tf co decir hex chara addr strin size_ ivale ivale	acter ess g t ence a [of * a co a v ole: If ence expr expr expr expr expr expr expr exp	<pre>s (c si c si c si c si c si c si c si c</pre>	har* tege 41 01 0	buf r CO decim nex octal chara null to null a ess tint* int int* int expr &exp f (ex)	<pre>5, i nsta nal cter ermin ddre ope ts * fi s * fi fi fi fi fi fi fi fi fi fi fi fi fi f</pre>	nt n ants hator ss rom if . then	, FI & ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<pre>LE* twis bit bit bit bit bit bit bit bit bit bit</pre>	stro e op wise o wise o wise o wise o wise o shift r shift l	eam) erat or and kor not right eft	FILE Ob10 Ob10 Ob10 Ob00	01 01 8 01 ^ 00001 00011: 00011: (& Ling 8 Examp	std: 0600 0600 111 = 11 >> 11 << () () () () () () () () () ()	in 11 == 11 == 2 == 2 == varia f a is f b is f c is 	 0b10: 0b10: 0b10: 0b000 0b001 0b001 x an in an in an in 	11 01 10 000 11100 dds * at at* at* at*	a "c "\ "\ ?:) si to it 1 1	ddre addre value write ther terr zeof	ess of ss of at a" v at a ope hary a - (*a) (<pre>> per v" a" rato 3>4 ? sized > 2 }) . 2 an ir an ir an ir += = /= = ^= </pre>	*ator *a ors 1:2 of(v) 4 4 4 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5	s &v *a = v == 2 == 4

 bow to write bug-free c DRY – Don't Repeat Yourse Learn to use your tools wel Fix "broken windows" (e.g., 	if • Get enough sle If • Plan before yo , warnings) • Crash early, e.	eep and nutrition. • Use assert(ou begin coding. • Free() where g., with assert(). • Design with o) to validate <i>your</i> code only. you malloc(), when possible. contracts.
how to debug			
 Test hypotheses systematic Take notes to stop going in Verify your assumptions. 	 Use the right deb circles. Write test code. Take a nap / wall 	bugging tool(s). • Trust the com • Do not trust S k / break. • Do not make r	piler. tack Overflow, friends, etc. random changes.
memory faults / Valgr	Ind error messages	"Conditional jump or move	"Definitely lest" leak
To start Valgrind, run: valgrind ./myprog	<pre>Segmentation fault - crash Writing at NULL with * int* a = NULL; *a = 10; Writing at NULL with -> Note the second se</pre>	<pre>conditional jump or move depends on uninitialised value(s)" If with uninitialized condition int a; // garbage!!! if(a == 0) {</pre>	Lose address of block void foo() { int* a = malloc(); } // !!!
"Invalid write"	<pre>Node a = NULL; a -> value = 10;</pre>	}	Lose address of address of block
<pre>Buffer overflow - heap int* a = malloc(</pre>	<pre>Writing at NULL with [] int* array = NULL; array[0] = 1; Reading from NULL with t</pre>	Loop with uninitialized condition int a; // garbage!!! while(a == 0) { //	<pre>void foo() { void** a = malloc(); *a = malloc(4); } // !!!</pre>
<pre>Write dangling pointer - heap int* a = malloc(); free(a); a[0] = 1;</pre>	<pre>int * a = NULL; int b = *a; Reading from NULL with -> Node* p = NULL;</pre>	<pre>Switch with uninitialized condition int a; // garbage!!! switch(a) { //</pre>	"Still reachable" – leak Address of block still in memory int main() {
"Invalid read"	<pre>int b = p -> value;</pre>	} Printing unterminated string	<pre>static void* a; a = malloc(); mature EVIT SUCCESS:</pre>
Buffer overread - heap int* a = malloc(4 * sizeof(*a)):	<pre>Reading from NULL with [] int* array = NULL; int b = array[0];</pre>	<pre>char s[2]; s[0] = 'A'; // no '\0' printf("%s", s);</pre>	<pre>"Invalid free()"</pre>
int b = a[10]; // !!!	Not detecting malloc() failure int* a = malloc("Use of uninitialized value"	"glibc free"
<pre>Read dangling pointer - heap int* a = malloc(4 * sizeof(*a)); free(a); int b = a[0]; // !!!</pre>	100000000000000000); *a = 1; // a is NULL Stack overflow	Passing uninitialized value to fn int a; printf("%d", a);	<pre>Double free int* a = malloc(); free(a); free(a); // !!!</pre>
Not detected by Valgrind	foo(); // !!!	"Syscall param uninitialised byte(s)"	<pre>Free something not malloc'd int a = 0; free(&a); // !!!</pre>
<pre>Buffer overread - stack int a[4]; int b = a[10]; // !!!</pre>	Writing to read-only memory char* s = "abc"; s[0] = 'A';	Return uninitialized value from fn void foo() { int a;	<pre>Free wrong part int* a = malloc(); free(a + 3); // !!!</pre>
<pre>Buffer overflow - stack int a[4]; a[10] = 1; // !!!</pre>	Calling va_arg too many times while (a == 0) { b = va_arg(); }	<pre>return a; } Write uninitialized value to file char c; fwrite(&c, 1, 3, stdout);</pre>	<pre>"silly arg () to malloc()" Negative size to malloc() void* a = malloc(-3); free(a);</pre>

© Copyright 2022 Alexander J. Quinn <aq@purdue.edu> except as noted. This content is protected and may not be shared, uploaded, or distributed. | Versions: This is v1.0.8 (8/22/2022) 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 © lan Foster.