Objectives - Thu 1/24/2019

- Type analysis exercise
- Call by reference
- GDB: whatis

pass by VALUE

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	int	a	10	locals	
224	int	r	13		
228	int	a	~~10~~ 5	args	callee-pbv
232	int	b	~~13~~ 7		
236					

address of instruction to execute after main returns

Heap

addr	value	
400		

Data segment

addr	type*	value
600		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume `sizeof(int)==4, sizeof(char)==1, sizeof(void*)==8.`

pass by ADDRESS

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	int	q	~~10~~ 5	locals	
224	int	r	~~13~~ 7		
228	int*	a	(220)	args	
236	int*	b	224		
244					

Heap

addr	value	
400		

destroyed (invalidated)

Data segment

addr	type*	value
600		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume `sizeof(int)==4`, `sizeof(char)==1`, `sizeof(void*)==8`.

Call by ADDRESS

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	int	eyy	~~10~~ → 5	locals	
224	int	bee	~~13~~ → 7		
228	int*	a	220		callee
236	int*	b	224		pass-by-addr
244	void*			ret addr	
252					

Heap

addr	value	🔒
400		

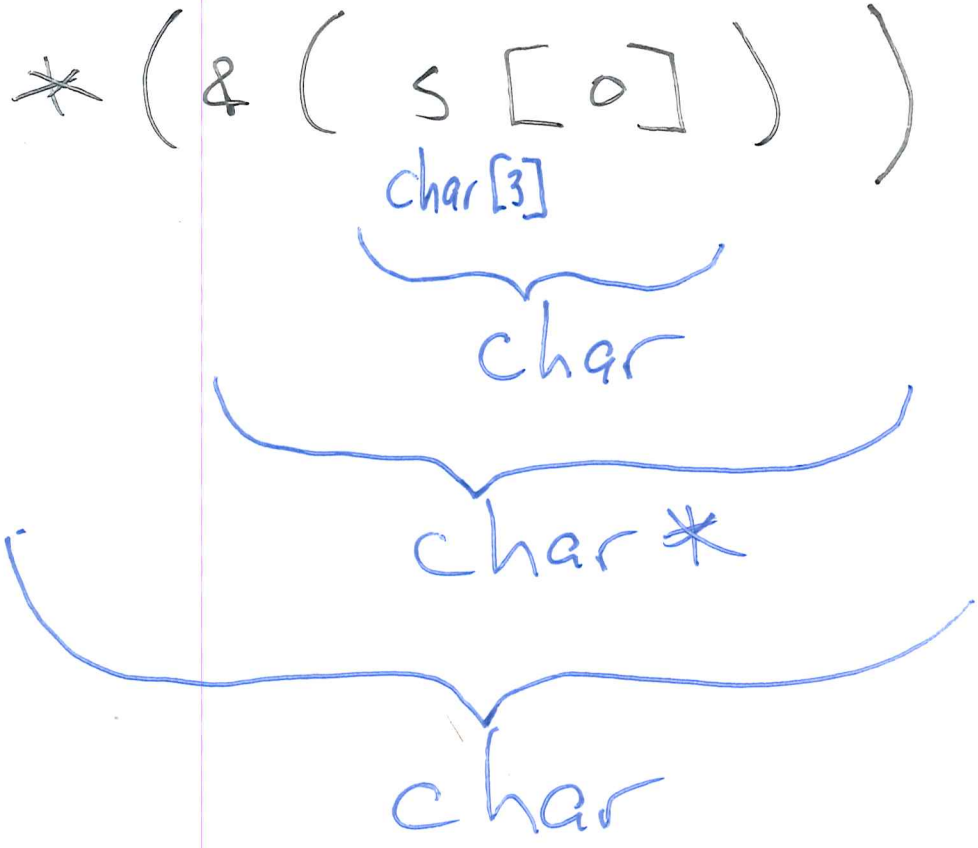
no locals in this fn

Data segment

addr	type*	value
600		

- Type and name are not actually stored in memory or executable. Addresses shown are fictional.
- Assume `sizeof(int)==4` `sizeof(char)==1` `sizeof(void*)==8`
- To show struct types with fields, split the type and name fields. In value field, just write the value of the field. Example →

type	name	value
Point : int,	p . x	5
: int	. y	6



*&s[0]

?

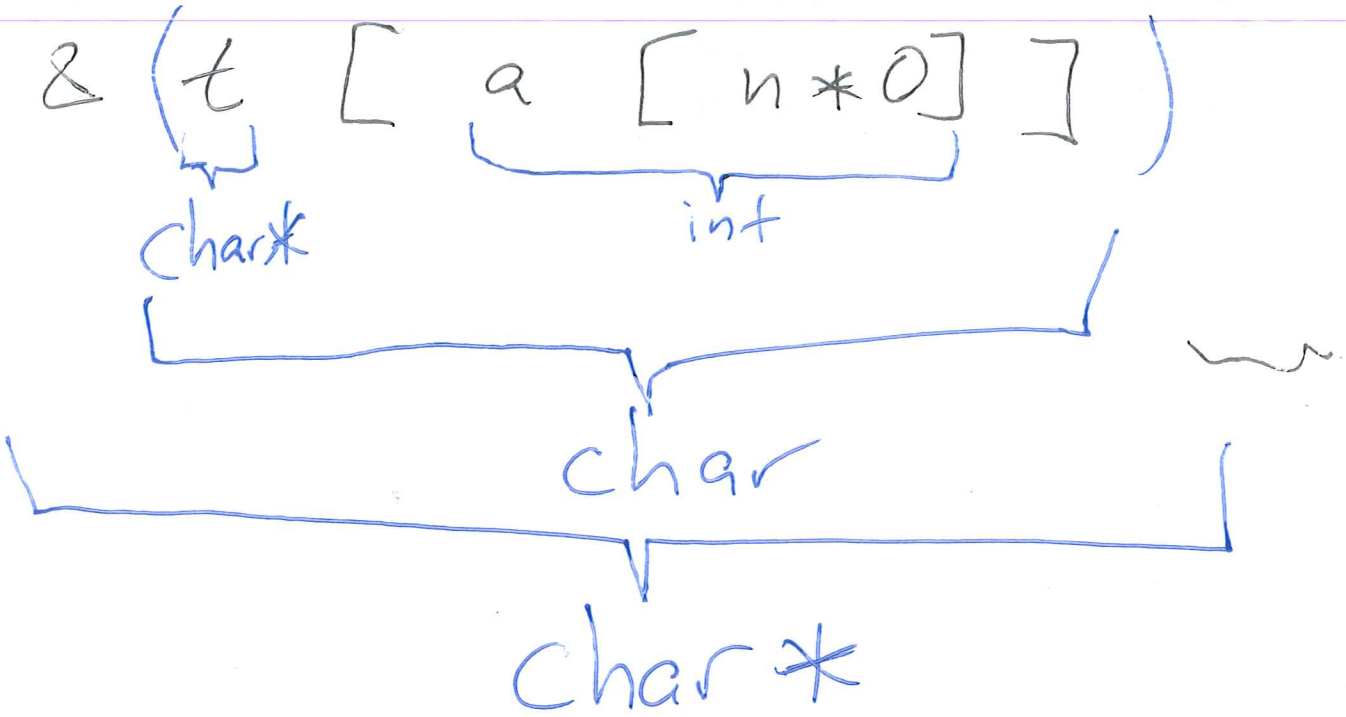
(*(&s))[0]

?

?

?

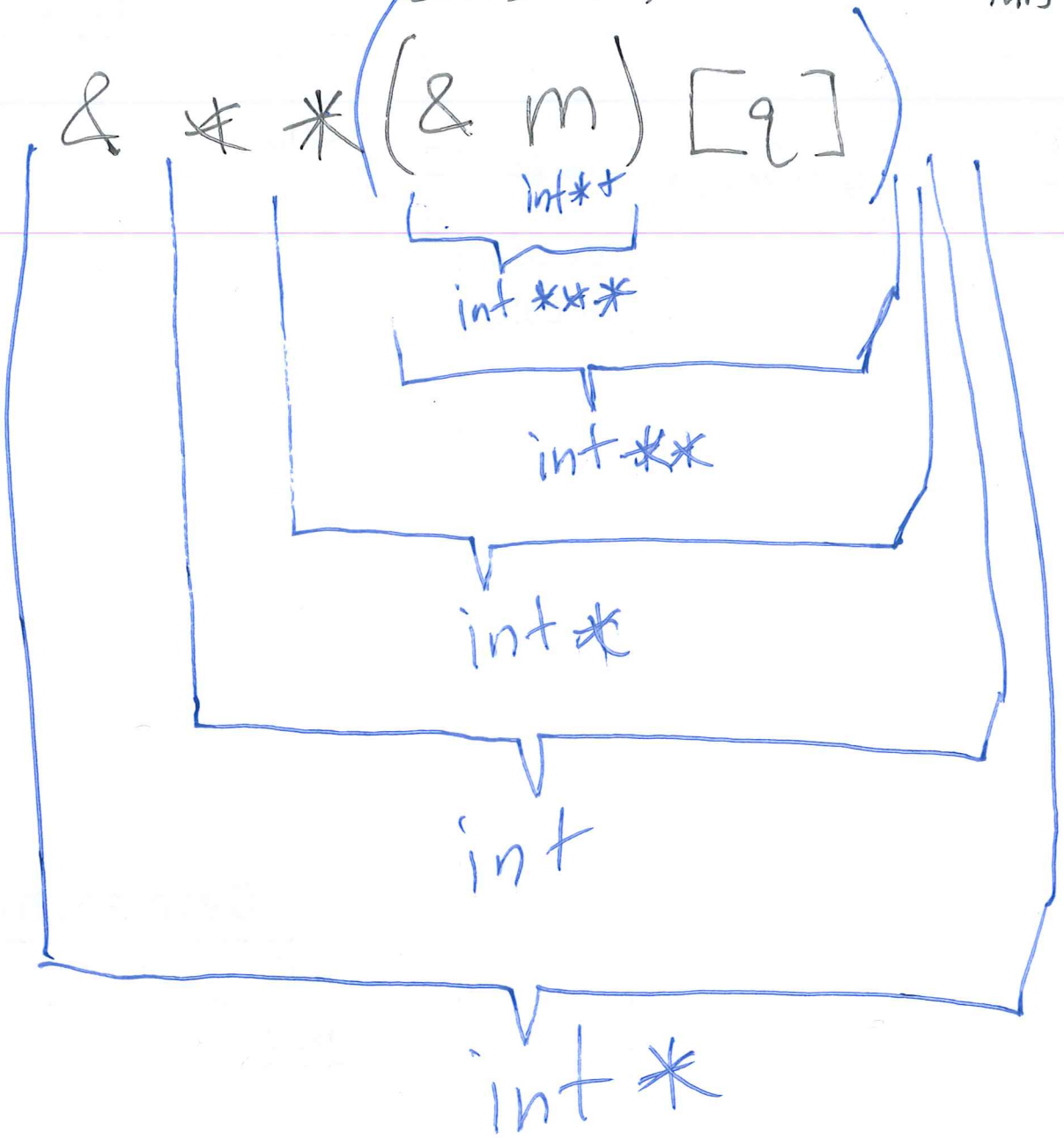
&t[a[n*0]]



while (true) {

int*** m = [redacted];

This code might not run.



Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	char*	s	600	locals	
228	char**	a_s	220		
236	char***	a_a_s	228		
	char**	aa	220		
	char*	a	600		

// vim: set ts=4 fenc=utf8 noet:

Heap

addr	value	
400		

Data segment

addr	type*	value
600	char*	"ECE"
604		

Type and name are not actually stored in memory or executable. Addresses shown are fictional. Assume `sizeof(int)==4`, `sizeof(char)==1`, `sizeof(void*)==8`.