Objectives - Fri 2/7/2020

- Call stack
- Pass-by-address
- swap(...) function

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	int	a	5	locals	
224	int	b	7		
228	int*	a_n1	220	args	swap(...)
236	int*	a_n2	224		
244	void*			ret addr	
252	int	temp	5	locals	
256					

This memory form diagram shows the state of the stack while executing swap_right.c, just before line 7 (i.e., after `int temp = *a_n1;` and before `*a_n1 = *a_n2;`).

The heap segment and data segment are not used.

The swap_right.c example was used in Prof. Quinn's lecture on 2/7/2020.

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`.

swap-wrong.c

This shows the memory state as of line 9 (just before swap(...) returns)

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	int	a	5	locals	
	int	b	7		
	int	n1	~~5~~ 7	args	swap(...)
	int	n2	~~7~~ 5		
	void*			ret addr	
	int	temp	5	locals	

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`
- 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

2-7-2020

swap_right.c

This shows the memory state as of line 9 (just before swap(..) returns).

Stack

addr	type*	name*	value	part	fn
200	int	argc	1	args	main(...)
204	char**	argv	→ {"./foo"}		
212	void*			ret addr	
220	int	a	~~5~~ → 7	locals	
224	int	b	~~4~~ → 5		
228	int*	a_n1	220	args	swap(...)
236	int*	a_n2	224		
244	void*			ret. addr	
252	int	temp	5	locals	

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`
- 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

0202-L-2