① Lecture-5

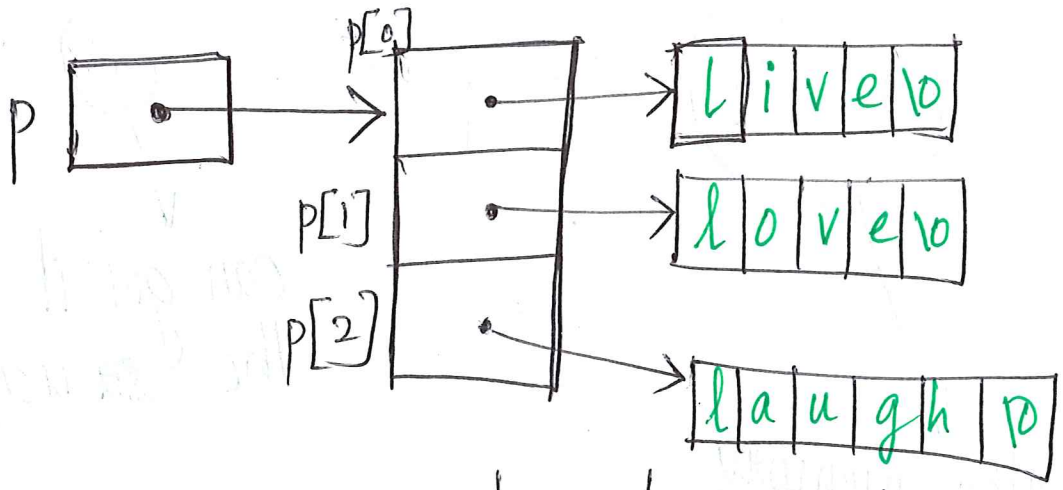
2d-arrays vs pointers to pointers

```
char a[3][10] = {{"life"}, {"play"}, {"work"}};
```

a:	0	1	2	3	4	...	9
0	l	i	f	e		...	
1	p	l	a	y		...	
2	w	o	r	k		...	

wasted.

```
char *p[] = {"live", "love", "laugh"};
```



p[i] - is a pointer to a char *.

p - is a pointer to an array of char * pointers

- is also a pointer to a pointer.

eg. `char **pp = p;` // assinging p to pp.

p and pp will point to the same thing. (ie) p[0].

Disadvantages of array. (2)

1. fixed size.
2. wastage of space (if contents is less than size).
3. insertion at the beginning - hard.

Dynamic memory allocation.

```
int *pm = malloc(sizeof(int) * n);
```

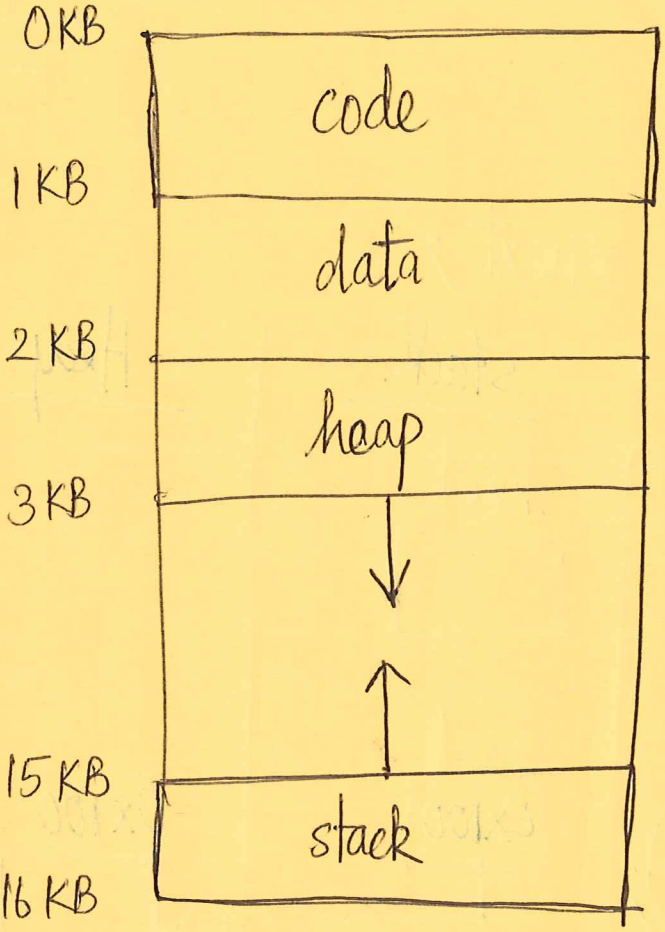
malloc → allocates memory in the heap

n → can get it from the user.

(*) Discuss about stack memory vs heap memory.

Memory of a process

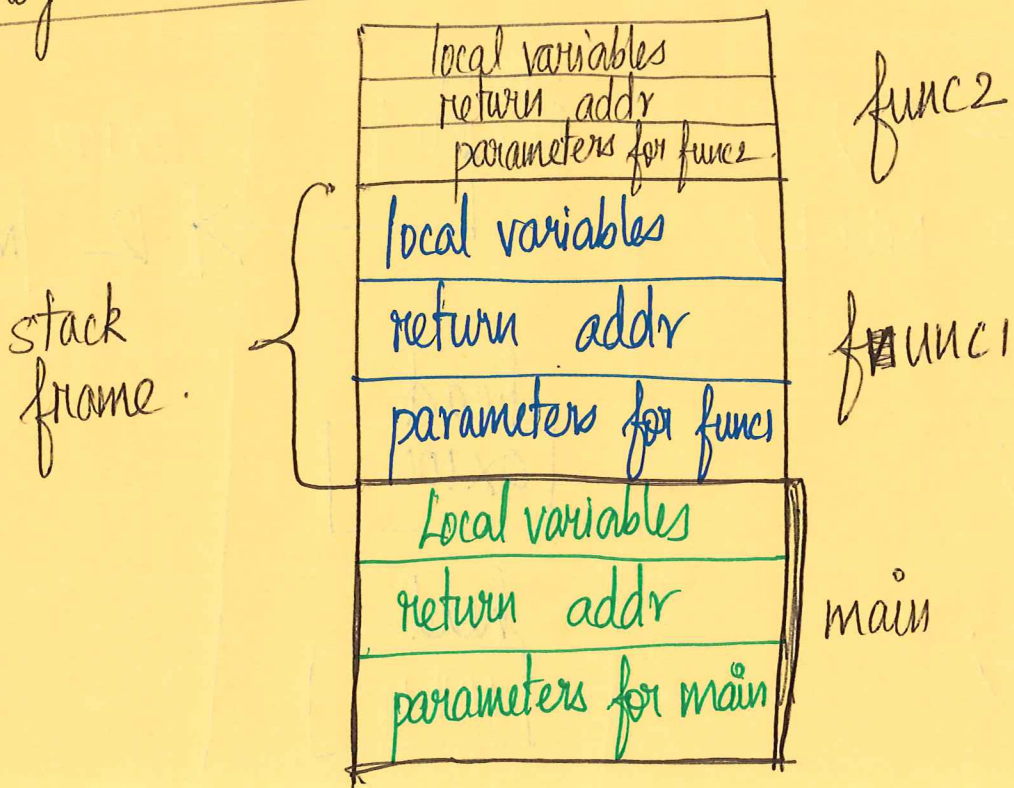
Ref: OSTEP by Arpaci-Dusseau



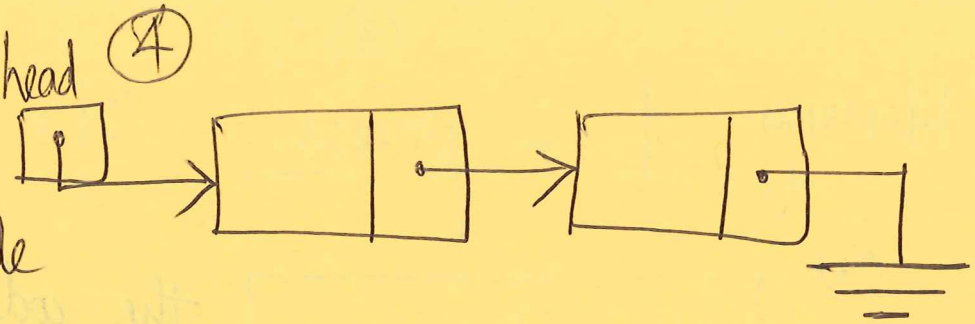
the code segment: where instructions life.

the data segment: contains global data.

Program stack



Linked Lists




```
struct node
{
    int data;
    struct node *next;
};
```

	stack	Heap
`struct node *head;`	head	
`head = malloc(sizeof(struct node));`	head 0x100	0x100 data next
`head->data = 12;` `head->next = NULL;`	head 0x100	0x100 12 NULL
`free(head);`	head 0x100	
`head = NULL;`	head NULL	