2D arrays vs pointers to pointers

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

Wasted.

char * p[3] = {"live", "love", "laugh"};

p[0] - is a pointer to a char *
p[1] - is a pointer to an array of char * pointers
p[2] - is also a pointer to a pointer.

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

p and pp will point to the same thing. (i.e) p[0].
Disadvantages of array.

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

Dynamic memory allocation.

\[
\text{int } * \text{pm} = \text{malloc} (\text{sizeof (int) } \times n);
\]

allocated memory in the heap

can get it from the user.

Discuss about stack memory vs heap memory.
Memory of a process

Ref: OSTEP by Attucci Dusseau

the code segment: where instructions live.

the data segment: contains global data.

Program stack

Stack frame:

local variables
return addr
parameters for func

local variables
return addr
parameters for func

Local variables
return addr
parameters for main

main

func

func2
Linked Lists

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

struct node *head;

head = malloc(sizeof(struct node));

head->data = 12;
head->next = NULL;

free(head);

head = NULL;