* doc - collection of paragraphs ⟶ lines ⟶ words ⟶ (a-z), (A-Z)

* convert document into paragraph, Sentence & words. ⟶ return specific Word, Para, Sentence, word!

char* ⟶ word "Hello\0"

char** ⟶ Collection of words — Sentence (" ") — Sepera (space)

char*** ⟶ " of sentences. (".") (Full Stop) — seperation.

char**** ⟶ paragraphs ⟶ Seperated by ("\n")

when last para ⟶ no newline!

Learning C is fun.
Learning pointers is more fun. It is good to have pointers.
] 2 paragraphs.

```
char ** first_sent_in_1_Para = {"Learning", "C", "is", "fun"};
char *** 1_Para_2_sent = {"It", "is", . . . . , "pointers"};
char *** Second Para = {{"Learning", . . . , "fun"}, {"It", . . . "pointers"}};
char **** document = {{{"Learning", "C", "is", "fun"}}, {{"Learning", . . . 
        "fun"}, {"It", . . . , "pointers"}}};
```

```
char **** get_document (char* text);
char *** kth_Para (char**** doc, int k);
char ** kth_Sent_m_Para (char**** doc, int k, int m);
char * kthword_mth ~~Sent~~_ nth_Para (char**** docu, int k, int m, int n);
```

* memset (ptr, '.', numBytes); ⟶ Fill with '.' in numBytes say: (12 bytes)

* char ****

pointer1 of pointer2 of pointer3 of pointer4 pointing to char.

word
pointer4 ⟶ char

words
pointer 3 ⟶ char *

Sent
pointer 2 ⟶ char **

Para
pointer 1 ⟶ char ***

Basic rule

- ★ space ' ' → next word of same para, sent
- ★ '.' → next sentence. (If '\n' not next) – else
- ★ '\n' → next paragraph

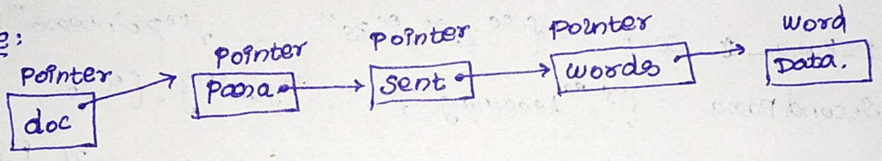
How to know end: '\0'

I/p: char* → get_document

"Learning C is fun.\nLearning pointer is more fun. It is good to have pointers.\0"

Grow approach

Basic structure:



★ dereference doc : which para (→ pointer to 0th sentence)

★ dereference : para : which sentence → (pointer to 0th words)

★ dereference : words : which word

what I get

grow approach

{{char* "}

{"Learning . . . , ... \0"}; → 1 hop to data

{{{"Learning", "C", ...}},{{"Learning", "pointers"}, {...}}};

1 hop → para (pointer pointing sent)

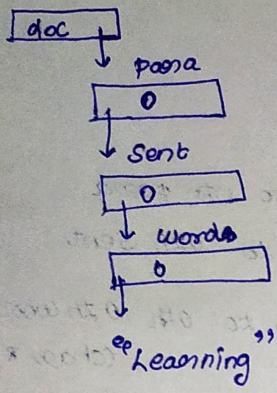
2hop → sentence (pointer pointing words)

3hop → words (pointer pointing word)!

4hop → Character data!

grow approach

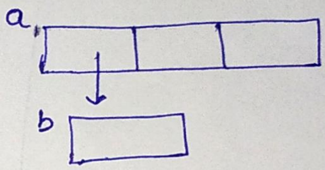
doc: pointer - Not array pointer.



whenever space: word - realloc

'.' : realloc sentence

'\n' : realloc para.



To realloc b

realloc (*a, 8 bytes)

realloc (pointer, size)

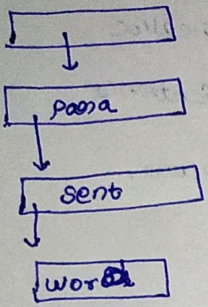
*a → gives address of (index - 0th) b

Kthword → index → K-1 (starting from 0)

```
① char * kth_word_in_mth_sent_n_para(char *** doc, int k, ~~int~~ m, int n)
  {
    return (doc [n-1][m-1][k-1]);
  }

② char ** k_~~Sent~~_m_para (char *** doc, int k, int m)
  {
    return (doc [m-1][k-1]);
  }

③ char *** kth_Para (char ***doc, int k){
    return (doc [k-1]);
  }
```

④ get char * $\xrightarrow{\text{return}}$ char *** approach: not going to copy anything – only pointer business!

doc-pointer(not array pointer).



why?

Actually: only one doc!

doc[0] → pointer pointing to 0th para

doc[0][0] → pointer pointing to 0th sent

doc[0][0][0] → pointer pointing to 0th 0th word (char*)

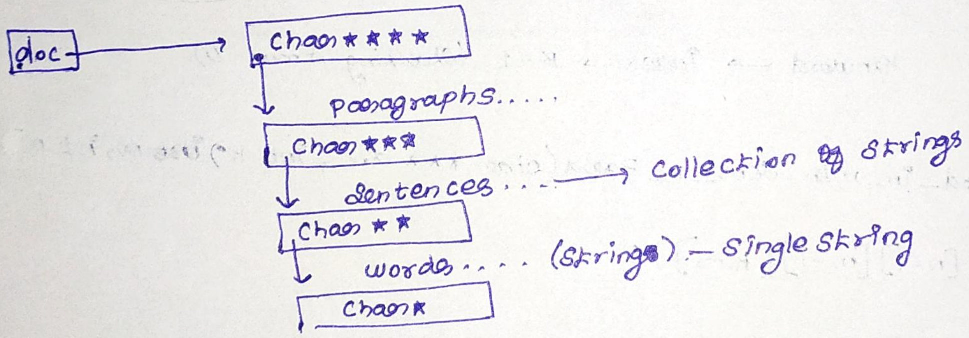
Char**(doc[0][0])

Char***(doc[0])

~~char~~ char****(doc)

*doc → which doc? (only one) → so 0th doc!

*


doc
char****
paragraphs....
char***
sentences... → collection of strings
char**
words.... (strings) – single string
char*

Learn C.
Learn. Learn C] a para.

i/p = "Learn C.\nLearn. Learn C \0";

0th para
0th sent
0th word = input (address of 0th index L)
when ' ' → \0 [indicate end of string: word]
word → 1st index

0th P
S
1st word = &input[6];
full stop and \n

Note: when ever full stop & \n → make

Full stop \0

So when ever "\n"

do '.' (word=0) → alloc one sent
do ' ' (word+=1) → alloc one word ← word=0

* i++ (ignore \n)
* make paragraph new, sentence =0

simple

\n → realloc para

that para → has no new Sent/word

So realloc → sent
realloc → word

execute: '.', ' ' blocks!

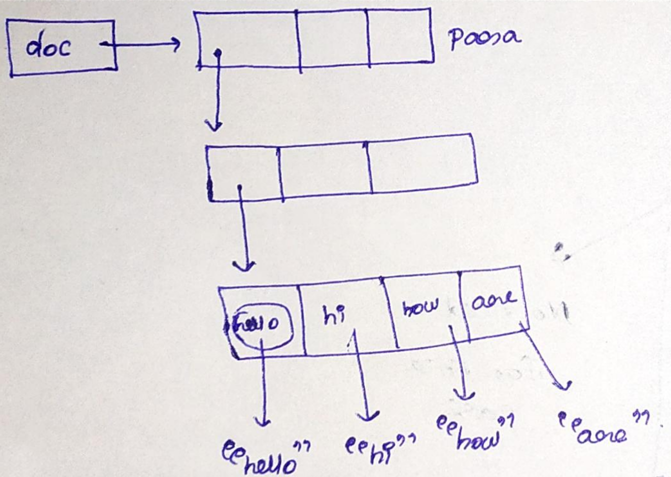
By making words, sents = -1.

So 0 th index, words, sent created.

'.' → execute words realloc do.

Note! : Switch Statement — when 1 case true all are true (cases): until break!

(end picture)



doc [0] → 0 th para
↓
points to 0th sentence

doc [0] [0] → 0 th sentence
↓
points to 0th word.
(pointer of word).

doc [0] [0] [0] → 0th word (pointer to word).
dereferencing
character.

\n

para++;

doc [para-1] = (char***) realloc → wrong!

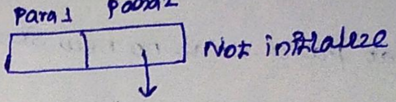
why doc has pointer to para
not! doc [para-1]
↓
has pointer to sentence!

```
for (i=0;        ; i++)
{
    if (text[i+1] == '\n')
    {
        text[i] = '\0';
        i++;
    }

    switch (text[i])
    {
        case '\n':


        .\0 [end]
```

\n

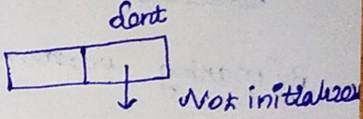
→* para++

→* reallocate



Initialize- sentences.

Initialize- words.



initialize!

make words = 0

Then initialize (1).

(i+1) ≠ '\n'

when '.'

* create new sentence
* word = 0

goto ' '

* create word (initialize)
* make '.' ↔ zero!

No need for this case

So run up to text[i+1] != '\0'