2510 : Lecture 20 March 08/17

Special Cases

```
realloc(p, Ø) ≡ free( p) /* realloc returns unlike free */
realloc(Ø, size) ≡ malloc(size)
```

ex. storing lines of text in 3 ways

① Using a 2D array of characters

```
#define LINESIZE 128
#define NLINES 1000

char lines[NLINES][LINESIZE];
size_t i, j;
for (i=0; i<NLINES; i++)
    if (!fgets(lines[i], LINESIZE, stdin))
        break;
for (j=Ø; j<i; j++)
    printf("%s", lines[j]);
```

lines is an array of NLINES objects each an array of LINESIZE chars

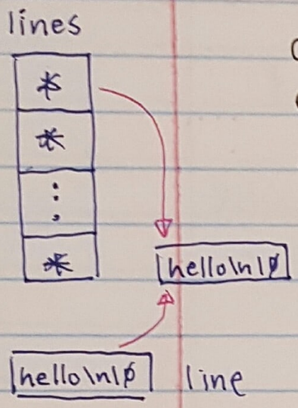
two disadvantages:

1) Wasting memory, max number lines is fixed
↳ most line don't need 128
2) NLINES IS FIXED

*② will slove disadvantge #2.

... Lecture 20||

ex. Storing lines of text :
② using an array of pointers to dynamic memory



```
char * lines [NLINES];     array of pointers
char line [LINESIZE];
size_t i, j;
for (i = 0; i < NLINES; i++){
    if (! fgets (line, LINESIZE, stdin)){
        clearerr (stdin);
        break;                        for null char
    }
    lines [i] = malloc (strlen (line) + 1 );
    if (lines [i] == 0){
        fprintf(stderr, "unable to allocate memory \n");
        break;
    }
    strcpy (lines[i], line);
}
```

how we use this :

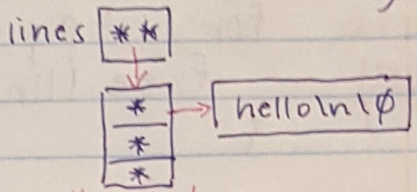
```
for (j = 0; j < i; j++){
    printf ("%s", lines [i] );
} /* deallocate the memory */

for (j = 0; j < i; j++)
    free (lines[j]);
```

disadvantage :
① max number of lines is still fixed

... Lecture 20||

ex. Storing lines of text

③ using a dynamic array of pointers each pointing to dynamically-allocated memory



What is the allocation strategy?

```
#define BLOCK 3
char ** lines;
char    line [LINESIZE];
size_t  nalloc; # of pointers allocated
size_t  nused; # of pointers used
size_t  i;
nalloc = nused = 0; → lines = 0;
while (fgets (line, LINESIZE, stdin)){   as long as we can read a line
    if (nused == nalloc){ if we have used all allocate ptrs
        char ** temp = realloc (lines, (nalloc + BLOCK)
                       times → * size of (char *));

        if (temp == 0){
          fprintf (stderr, "fail");
          break;
        }
        /* succeeded in getting ~~reading~~ more pointers */
        lines = temp;
        nalloc += BLOCK;
    }
    lines [nused] = malloc (strlen (line) + 1);
    if (lines [nused] == 0){
        fprintf (stderr, "...");
        break;
    }
    strcpy (lines [nused ++], line);
}
```

continued...

... Lecture 20

continued...

```
for (i=0; i<nused; i++){
    printf("%s", lines[i]);
}
```

deallocate memory

```
for (i=0; i<nused; i++){
    free(lines[i]);   deallocate each line
}

free(lines);   deallocate array of ptrs
```