- stdio:

Based on file streams (FILE *)

```
FILE *fopen(const char *path, const char *how)
int   fclose(FILE *file);
int   fgetc(FILE *file);
int   fputc(int c, FILE *file);
```

- Macros → text subtitutions ← useful because "functions" do not need types (can use int & float)

```
#define SIZE 32
```

e.g.
```
int count[SIZE];
:
for(i=0; i<SIZE; i++)
    count[i]=0;
```

↳ #define SIXTEEN 16

```
#define min(a,b) ((a<b) ? a:b))
```
cond ? true : false

```
min(x++, j++)
```
variable gets incremented twice

↳ ((x++ < y++) ? x++ : y++)

```
#define square(a) ((a)*(a))
```
e.g. res=square(5)
↳ 5*5

res=square(2+3) -> 2+3*2+3

gcc -E ← just preprocess

Conditional Compilation

```
#ifdef  symbol
#endif
#ifdef  DEBUG
#fprintf(stderr, ...);
#endif
```

foo.h:

```
#ifndef FOOH
#define FOOH
    :
    :
#endif
```

Arrays as Parameters:

↳ passing location by value

↳ two ways

- type *parameter
- type parameter[]

```
int main(int args, char *argv[])
{
 int i;
 for(i=0; i<argc; i++)
 {
   printf("%s ", argv[i]);
 }
 putchar('\n');
 return 0;
}
```

```
$./echo a b c
 a b c␣
```

```
int main(int args, char *argv[])
{
 int i;
 for(i=0; i<argc; i++) {
  if(i<argc-1) printf("%s␣", argv[i]);
  else printf("%s", argv[i]);
 putchar('\n');
 }
 return 0;
```

```
$./echo a b c
 a b c
$
```

```
int main(int args, char *argv[])
{
 int i;
 for(i=0; i<argc; i++) {
  printf("%s %c", argv[i], (i<argc-1)?' ':'\n');
 putchar('\n');
 }
 return 0;
```

```
$./echo a b c
 a b c
$
```

ERRNO

void perror(const char *msg);

```
FILE *in;

in = fopen("file", "r");
if(in == NULL) { perror("file");
    exit(EXIT_FAILURE);
}
```