

read file char-by-char

```
#include <stdio.h>

FILE *h;
int ch;
h = fopen("filename", "rb");
/* error checking missing */
while ((ch = fgetc(h)) != EOF) {
    /* deal with ch */
}
/* if needed test why last read failed */
if (feof(h) || ferror(h)) /* whatever */;
fclose(h);
```

You can replace fgetc(h) with getchar() to read from standard input.

read file line-by-line

```
#include <stdio.h>
FILE *h;
char line[100];
h = fopen("filename", "rb");
/* error checking missing */
while (fgets(line, sizeof line, h)) {
    /* deal with line */
}
/* if needed test why last read failed */
if (feof(h) || ferror(h)) /* whatever */;
fclose(h);
```

Flexible Array Member

How to declare a FAM?

By using empty brackets as the last member of a struct.

How to define the size for an object containing a FAM?

```
ptr = malloc(sizeof *ptr + sizeof (FAMTY-
PE[wantedsize]));
```

Do not use FAMs! They were known as *struct hack* before C99 and, now as then, feel like a dirty hack.

<stdio.h> functions with a FILE pointer at the end

```
char *fgets(char *, int, FILE *);
int fputs(int, FILE *);
int fputs(char *, FILE *);
size_t fread(void *, size_t, size_t, FILE *);
FILE *freopen(char *, char *, FILE *);
size_t fwrite(void *, size_t, size_t, FILE *);
int ungetc(int, FILE *);
```

dynamic memory

Remember to **#include <stdlib.h>**

Allocate

```
malloc ptr = malloc(n * sizeof *ptr);
calloc ptr = calloc(n, sizeof *ptr);
```

Change size

```
realloc newsize = n * sizeof *ptr; tmp =
realloc(ptr, newsize); if (tmp) ptr
= tmp; else /* ptr is still valid */;
```

Release

```
free free(ptr);
```

remove trailing newline

How do I remove the final newline in a string?

```
len = strlen(data);
if (len && data[len - 1] == '\n') data[--len] =
0;
```

or, if you don't need to keep and update data length

```
data[strcspn(data, "\n")] = 0;
```

If len is known in advance, do not call strlen(). You can pass the updated len to the caller.

Casting

Casts in C are almost always wrong. When are they right?

```
<ctype.h> isupper((unsigned
char)ch)
%p printf printf("%p", (void*)ptr)
specifiers
```

Specifically a cast to the return value of **malloc()** is a definite sign the code author either didn't know what he was doing or didn't choose a good language for the implementation of whatever he's doing.

(BSD) sockets

Headers needed

```
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
```

initialize with

```
getaddrinfo()
```

loop to find and connect a socket

```
socket()
```

```
connect()
```

if needed: close()

```
after loop: freeaddrinfo()
```

```
getpeername(), getsockname()
```

```
send() or recv() or sendto() or recvfrom()
```

```
close()
```

Predefined C macros

```
__FILE__
```

"filename.c" or something like that

```
__LINE__
```

42 or another integer

Predefined C macros (cont)

__STDC__

1

__STDC_VERSION__

undefined for C89; **199901L** for C99; **201112L** for C11

__DATE__

"Feb 17 2012" for example

__TIME__

"15:16:17" for example

__func__

"main" for example

__STDC_HOSTED__

0 or 1

Reserved identifiers

Reserved for all uses anywhere

_[A-Z]*; __* E[A-Z]*; E[0-9]*

is[a-z]*; to[a-z]* SIG[A-Z]*; SIG_[A-Z]*

LC_[A-Z]* *_t

str[a-z]*; mem[a-z]*; wcs[a-z]*

all math functions possibly followed by **f** or **l**

When **#include <limits.h>** is present

*_MAX

When **#include <signal.h>** is present

SA_* sa_*

POSIX adds a few other identifiers

<dirent.h> d_*

<fcntl.h> l_*; F_*; O_*; S_*

<grp.h> gr_*

<pwd.h> pw_*

<sys/stat.h> st_*; S_*

<sys/times.h> tms_*

<termios.h> C_*; V_*; I_*; O_*; TC*; B[0-9]*

