Input and Output [Lecture 2]

1. Standard I/O [K&R 7.1]

```
int getchar(void);
int putchar(int);
```

* #code1 - upper.c

2. Formatted Output - printf [K&R 7.2]

```
int printf(char *format, arg1, arg2, ...);
int sprintf(char *string, char *format, arg1, arg2, ...);
```

3. Formatted Input - scanf [K&R 7.4]

```
int scanf(char *format, ...);
```

(int) → returns no. of items successfully matched.

```
int sscanf(char *string, char *format, arg1, arg2, ...);
```

Args to scanf - pointers!

(X) The arguments to scanf and sscanf must be pointers!

```
scanf ("%d", n);
scanf ("%d", &n);
```

#code2 - fun_with_scanf ;

4. File Access [R & R 7.5]

standard file descriptors — 0, 1, 2
- 0 → stdin
- 1 → stdout
- 2 → stderr.

1) Open a file (fopen)

```
FILE *fp;
fp = fopen (name, mode);
```

- name: char [] file name
- mode: char string "r", "w", "a".

"b" - appended to mode if file is a binary file.

2) Check for errors.

Admin

1. Waitlist.
2. Piazza.
3. OH (tentative)
4. Po - clarification.
5. Linux

3) Read / Write to a file.

4) Close the file.

5. Line I/O [K&R 7.7]

```
char *fgets(char *line, int maxline, FILE *fp);
int fputs(char *line, FILE *fp);
```

The UNIX System Interface [K&R 8.1]

1) File Descriptors

fd - a small non-negative integer.

0 - stdin
1 - stdout
2 - stderr

(X) prog < infile > outfile

2) Low Level I/O – Read and Write [K & R 8.2]

```
int n_read = read(int fd, char *buf, int n);
int n_written = write(int fd, char *buf, int n);
```

Limit of no. of files a program may open = 20 (usually)

no. of bytes read / written (1 or 1024 or 4096 bytes.)

3) Open, creat, close [K&R 8.3]

```
int open(char *name, int flags, int perms);
```

flags: O_RDONLY, O_WRONLY, O_RDWR

perms: 0

```
int creat(char *name, int perms);
int close(int fd);
```