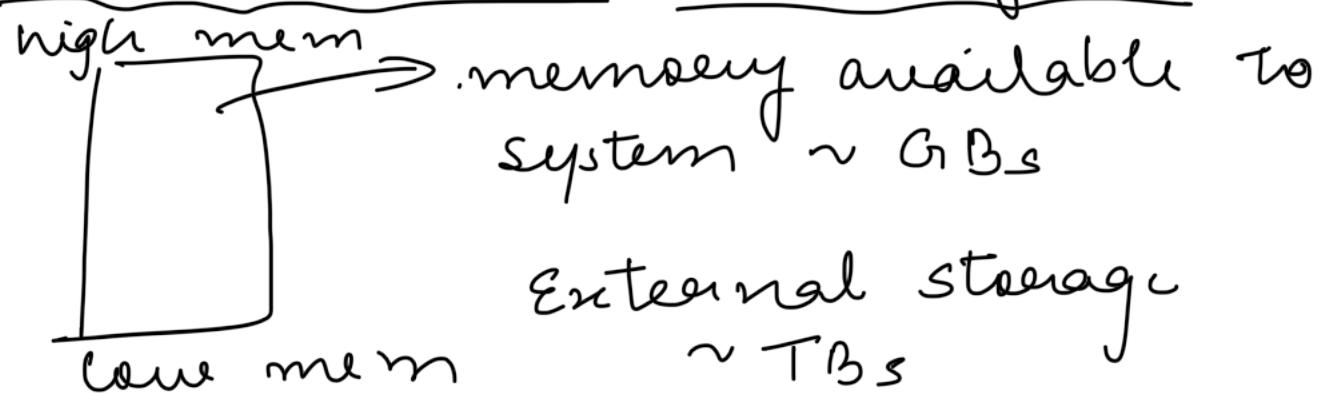


C-Week-10 - Notes

1) Introduce " to File Handling in C



- storage -

1. on-chip registers in CPU → fastest, smallest ~ KBs
2. Dynamic RAM → slower ^{than}, ~ GBs
3. External storage → slowest, ~ TBs

- persistence - across power cycles

- structured storage → file sys.

1. hierarchy - indexable / findable storage
2. named file - store arbitrary data
3. organize into folders - possible

- file → name, path on sys., arbitrary data, etc.

- programming interface - sys. calls

→ direct hardware access not allowed
→ go through OS kernel

- FILE → struct

- FILE* → fopen / fclose (pointers to file struct), permissions, disk space, paths, many possible errors

- special files → `stdin`, `stdout`, `stderr`

```
if ((fp = fopen ("test.txt", "u")) == NULL)
{
    _____
}
```

L2 File Handling in Text Mode - Part I

- text mode - human readable files, ASCII / unicode encoding (display on screen), no. as digits
- `fgetc`, `fgets`, `scanf`
 $fgetc(\dots, \text{stdin}) \equiv \text{gets}(\dots)$
- `fprintf`, `ffputs`, `ffprintf`
 $ffputs(\dots, \text{stdout}) \equiv \text{puts}(\dots)$
- no equivalent for `stderror`

```
while ((ch = fgetc(fp)) != EOF)
{
    _____
}

while (fgets(s, 1024, fp) != NULL)
{
    _____
}
```

```
    }scanf(fp, "%s", s)
    fgets(s, 1024, fp)
```

L3 Part -2

```
fputc('H', fp);
puts("Hello", fp);
```

- std::in, std::out works similar like Linux systems.

L4 File Handling in Binary Mode

- binary mode - some OS handle text separately, OS filters data before application, but can cause problems.
- it gives direct byte level access to data.
- ~~freefile~~ ~~freearray~~ \rightarrow direct map from file to array
- potential problem - indirection
hd - n 100 main

L5 Bit Manipulation

- bitwise operators

$\&$ (and), $|$ (or), \wedge (xor), \sim (NOT)

- masks

0010 1001 1100 \rightarrow LSB
 \hookrightarrow 9

- func \Rightarrow - check, Toggle, set, largest bit