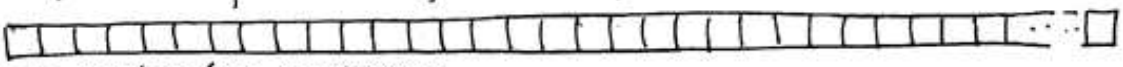
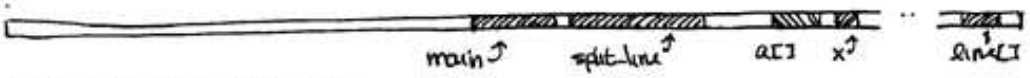
ARRAYS: INDEXING, POINTING

1 computer memory is an array of memory locations



- each location has an ADDRESS
- all the data (variables) and code for the program are stored in memory

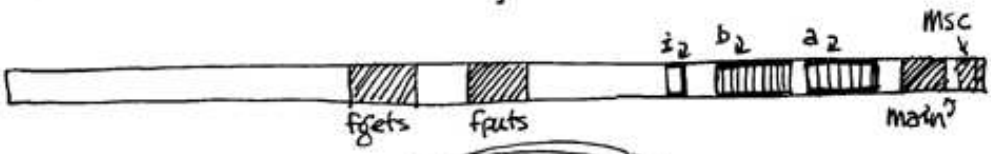


∴ every variable and every function has an ADDRESS

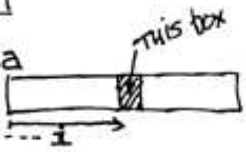
2 Understanding strcpy()

```
main()
{
    char a[20],b[20];
    fgets(a,20,stdin);
  mstrcpy(b,a);
    fputs(b,stdout);
}
```

```
mstrcpy(char d[],char s[])
{
    int i=0;
    while (s[i] != '\0'){
        d[i] = s[i];
        i++;
    }
    d[i]='\0';
}
```



copying a string



IDEAS
- index vs address
- array name == address
- base and offset
- passing an array to a function
- a[i] MEANS

EXAMPLE 2 sum( )
EXAMPLE 3 strchr()

3 PROGRAMMING WITH ADDRESSES

POINTER a variable that stores a memory address

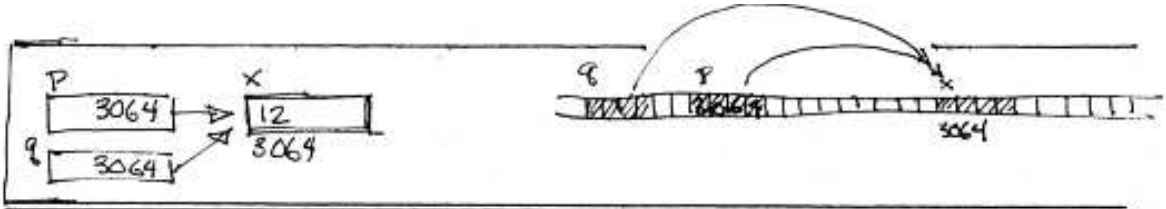
Declaring a pointer : `char *p; int *r; float *s;`

operations
- `&var`
- `*p`
- `p[n]`
- `p+n`

`p1 < p2`
`p1 == p2`
etc

EX copy2.c

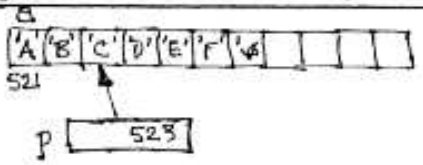
POINTERS BASICS



POINTERS and FUNCTIONS

```
main()
{
    int x;
    f(&x)
}
f(int *p)
{
    *p = 4;
}
```

POINTERS and ARRAYS




```
while (*p++)
    ;
len = p - a - 1;
```

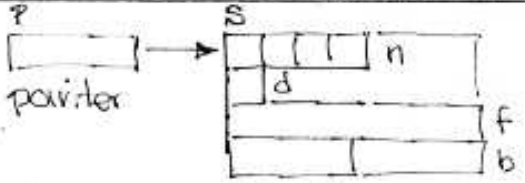
INDEXING and POINTING

```
a[2]    (a+3)[1]    *a      *(a+2)
p[2]    (p+1)[4]    *p      *(p+2)
```

ADDR[OFFSET] *(ADDR)

ADDR can be ① array name
② pointer value
③ &variable

POINTER TO A STRUCT



NOTATION

p→n	s.n
p→d	s.d
p→n[2]	s.n[2]
p→f	s.f

ARRAY OF POINTERS

```
char *a[4]
```