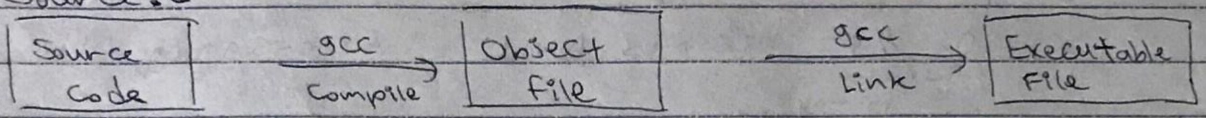
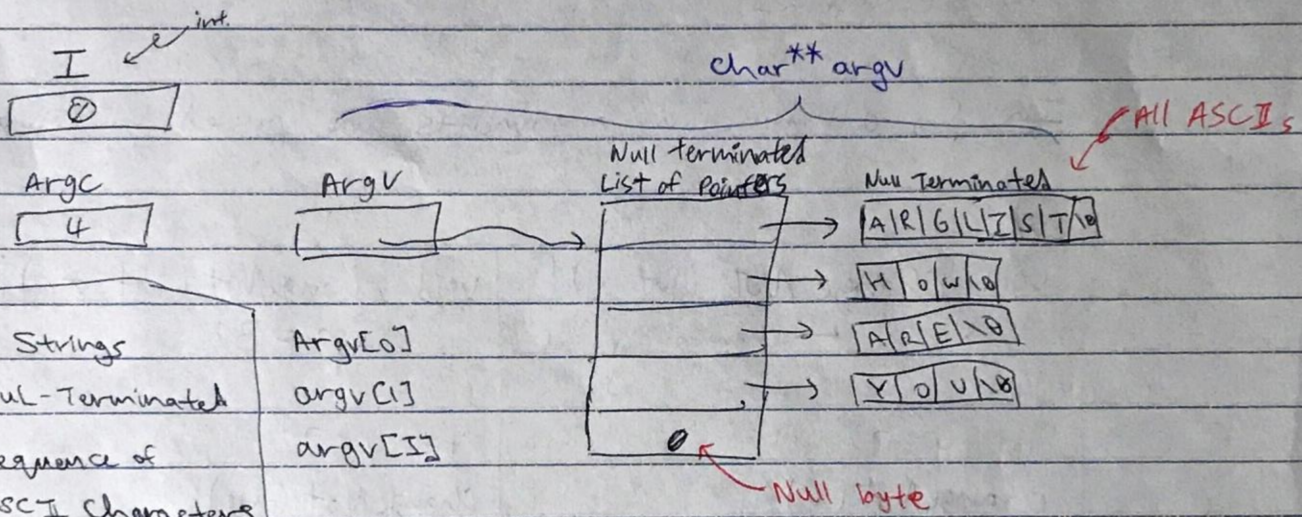
Header.H

Source.c



ASCII	-NOT Human Readable	-Not Human Readable
Human Readable	-Machine Specific	-Machine Specific
Portable (mostly)	-OS Specific	-OS Specific

07/03



* · C Strings
· NuL-Terminated
· Sequence of ASCII Characters

very Important!

C Pointers:

- A pointer contains an address "Pointer To" and "Address of" are equivalent terms.
- Use * in declaration of pointer variable
  - int x; // Type: int
  - int * px; // Type: Pointer to int

Type: char != char* != char** != char***

$\underbrace{px}_{\text{pointer to int}} = \overbrace{\&x}^{\text{pointer to int}}$

↖"address of"

```
int *p1, *p2, y;
p1 = p2;
p2 = 0; // p2 becomes NULL. (not assigning value of 0.)
     ↖ null
       Pointer
```

Dereference Pointer:

```
px = &x;
*x = 3; // now x==3
↳ * dereference
```

In C, all function arguments are passed by value.

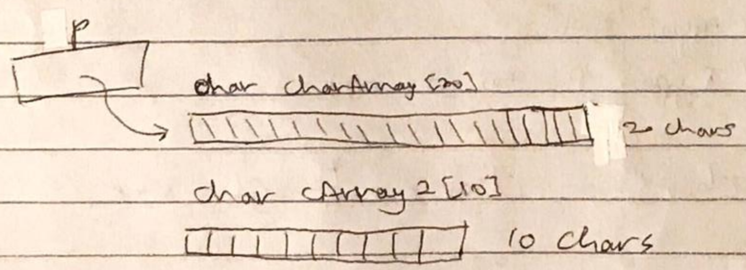
ex)

```
void Change NOT (int x){
    x=3; ← changes copy of y
}
int y=7;
changeNot(y);
// y does NOT change!
```

```
void changeNot (int *p){
    *p =3;
}
int y=7;
changeNot (&y);
// y DOES Change!
```

pass by reference VS pass by value

If char *p; char c; char charArray[20];
then, p=&c; p= charArray; p="HELLO";

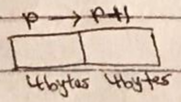


char *p

p= charArray

p= cArray2

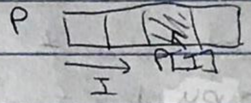
```
int *p;
++p      p++
--p      p--
```



Assignments:

```
*p                                   int p;
p1 = p2                              p = 0xC00420;
                                        ↕ 16 bytes = 10
p1 == p2        p1 == 0              p+4 = 0xC00430;
p1 != p2        p1 != 0
p += i   or   p -= i                 What we say      What it means
p + i   =   i + p                    p+1              Add size of (Int)
p1 - p2         p1 > p2   p1 >= p2   int *p           to p
                p1 < p2   p1 <= p2

*In C,   P[I]  =  I[p]
```



C Library String Functions:

```
#include   <String.H>
char *strcpy (char *DST, const char * SRC);
```

Promise not to change SRC string

```
    char Dest[20], Source[20]

    Strcpy (DEST, Source);
    Strcpy (Dest, "HELLO");

char *Strcat (char * DST, const char * SRC);
    char  Dest [10] = "ABC";      |A|B|C|\0|?|?|?|?|?|?|
    char  Source [] = "DEF";
    Strcat (Dest, Source);
    Dest |A|B|C|D|E|F|\0| | |
```

spaces preserved just for these characters.

*Always initialize

```
Void *GP;  int *PI;  char *PC;   // deref. of void * is bad.

GP = PI;       // can assign any type pointer to void *
GP = PC;

PC = GP;       // can pass any type pointer to void *
PI = GP;
```

Casting:

1. Change from

```
int x;
float y;

y = (float)x;
x = (int)y;
```

2. Shut Compiler up

```
int *PI;
char *PC;

PC = (char *) PI;
PI = (int *) PC;
```

```
//concat  * [ Char * Strncat (char * , const char *, size_T LEN)
//copy        char * strncpy (char *, const char *, size_T LEN)
```

copy at most LEN chars

```
//Compare   int strcmp (const char * S1, const char * S2);
```

↳ 0: S1 equals S2

<0: S1 less than S2

>0: S1 greater than S2

HW Assignment:

Functions

Binary ASCII ⟷ int

Decimal ASCII ⟷ int

octal ASCII ⟷ int

Hex ASCII ⟷ int

Pseudocode Binary ASCII to Int

```
Input is Binary ASCII String S
value = 0;    ← two's complement result.
Loop left to right thru String S, stop at NULL.
In each loop iteration I initial 0
    if Str[I] is '1', value = (Value << 1) + 1;
    else if Str[I] is '0', value = value << 1
    else ERROR (invalid input)
end loop
```

→ Build bit pattern

Pseudocode Hex ASCII to Int

Input is Hex ASCII String S

```
value = 0
Loop
  Switch (Str[I]){                                  left shift!
      case '0':  value = (value << 4);  break;
      case '1':  value = (value << 4) + 1;  break;
form                  :
bit pattern
      case 'a' || case 'A' :  value = (value << 4) + 10; break;
      default: invalid Hex ASCII
  }
end loop
```

*Decimal, keep track of '-' sign. Instead of shifting, times by 10.

Pseudocode Int to binary ASCII

Input unsigned int x, char * output

↖ large enough to hold output string

```
Loop left to right thru bits in X 64 times
   N = X >> 63   capture next high bit
   If N=0, put '0' in Str[I]
   else put '1' in Str[I]
    get next bit
end
Str[64] = '\0';
```

07/05 *Macro = text execution

makefile , (command + make) -i, -f, -n

↳ All the header files that are dependent will all get recompiled.

clean: does not depend on anything

```
  rm -rf *.o
```

↖ no blank! or All files will be deleted