CS246
Unix: the network
C: bits

April 26

FROM MONDAY
fwrite fixed

- Why not write the array of pointers??

- Problems remain??
  - UNFREED MEMORY

```c
#define SIZ 100
typedef struct
{
    int aa;
    double bb;
} aabb;
int main(int argc, char const *argv[])
{
    aabb **arr = malloc(SIZ * sizeof(aabb *));
    for (int i = 0; i < SIZ; i++) {
        arr[i] = malloc(1 * sizeof(aabb));
        arr[i]->aa = i;
        arr[i]->bb = sqrt(i);
    }
    FILE *fp = fopen("astrfp", "w");
    int d = SIZ;
    fwrite(&d, sizeof(int), 1, fp);
    for (int i = 0; i < SIZ; i++)
        fwrite(arr[i], sizeof(aabb), 1, fp);
    fclose(fp);
    return 0;
}
```

Lab

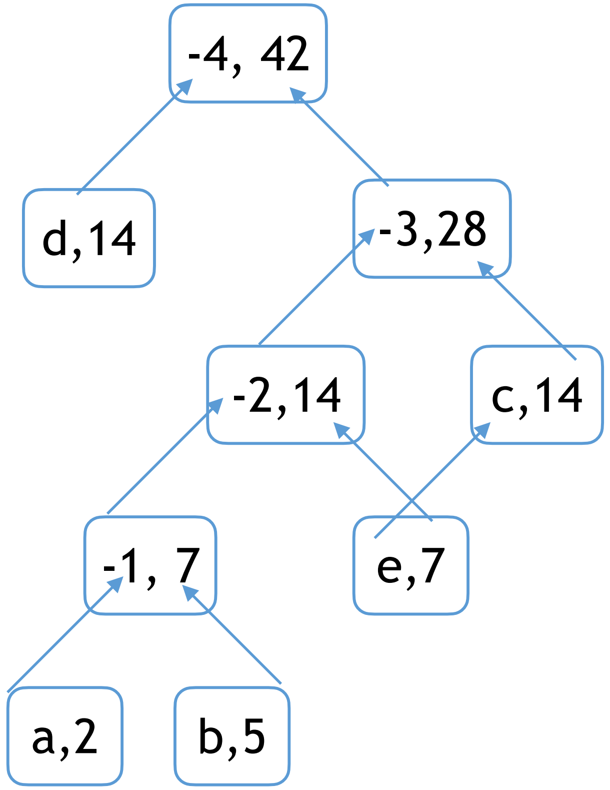
- Given the table at right
  - Construct a Huffman tree
  - Encode
    - abed
  - Decode
    - 101100110011010
- Send: tree, encoding and decoding

Character	Count
a	2
b	5
c	14
d	14
e	7

Hufmann Tree Building

- First item off PQ is left child, second item is right child
- Break ties in favor of lesser char/id.
    - when making new nodes, number them -1, -2, -3 …
    - use that number when breaking ties.

Char	Count	Code
a	2	1000
b	5	1001
e	7	101
c	14	11
d	14	0



printenv

- All of the “environment variables” set on machine
- printenv XXX
  - show value of that env var
    - printenv HOME
- echo $XXX
  - same as printenv XXX but more shell script like
    - echo $HOME

```
[gtowell@benz ~]> printenv
SHELL=/bin/bash
PWD=/home/gtowell
LOGNAME=gtowell
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/home/gtowell
LANG=en_US.UTF-8
LC_TERMINAL=iTerm2
SSH_CONNECTION=100.14.58.158 51932 165.106.10.169 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=gtowell
LC_TERMINAL_VERSION=3.4.5
DISPLAY=localhost:10.0
SHLVL=1
XDG_SESSION_ID=2447
XDG_RUNTIME_DIR=/run/user/10005
PS1=\e[0;36m[\u@\h \W]> \e[0m
SSH_CLIENT=100.14.58.158 51932 22
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=.:/home/gtowell/bin:/bin:/usr/local/sbin:/usr/local/bin:/usr/s
local/games:/snap/bin:/usr/bin
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/10005/bus
SSH_TTY=/dev/pts/0
 =/bin/printenv
```

Your Machine

- printenv HOST
  - probably returns nothing, on some systems, the hostname
- hostname -i -I -d

- 127.0.0.1/127.0.0.1 == localhost.
  - allows for the simplect possible network test — call yourself.

- IP address and DNS name
  - IP address uniquely identifies machine
    - sort of

```
[gtowell@benz ~]> cat /proc/version
Linux version 5.8.0-50-generic (buildd@lgw01-amd64-030)
(gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0, GNU ld (GNU
Binutils for Ubuntu) 2.34) #56~20.04.1-Ubuntu SMP Mon Apr
12 21:46:35 UTC 2021

[gtowell@benz ~]> hostname -d
cs.brynmawr.edu
[gtowell@benz ~]> hostname -i
127.0.1.1
[gtowell@benz ~]> hostname -I
165.106.10.169
[gtowell@benz ~]> hostname
benz

[gtowell@powerpuff ~]$ who
compsci  tty1         2020-03-03 18:02
gtowell  pts/4        2021-04-28 06:21 (100.14.58.158)
mvajpeyyii pts/13       2021-04-28 06:46 (165.106.118.210)
ddiaz1   pts/20       2021-04-19 10:51 (165.106.10.235)
gtowell  pts/19       2021-04-28 06:54 (100.14.58.158)
[gtowell@powerpuff ~]$ whoami
gtowell
```

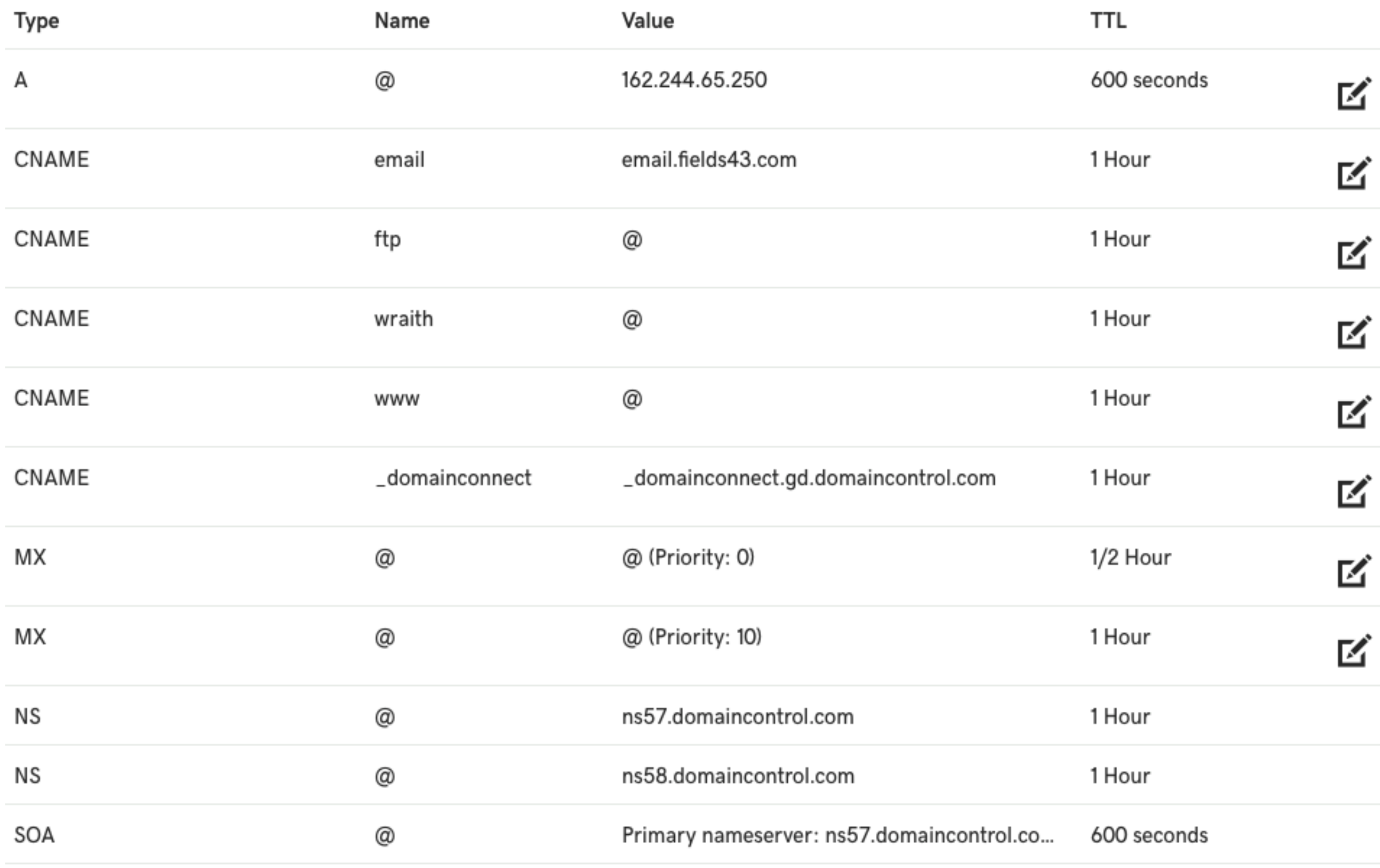
DNS

- Domain Name Service
  - translate a DNS name into an IP address
    - idea: you “register” a domain name with the appropriate agent. That agent then knows your IP address
  - “Agents” exists for each “.” separated part of a name.
  - Root level … .com, .edu …
    - so Bryn Mawr had to register the name brynmawr with the agent for .edu creating brynmawr.edu
  - Then the CS department had to register the name cs with Bryn Mawr, thereby creating “cs.brynmawr.edu"
  - Then to create name powerpuff.cs.brynmawr.edu …

More DNS

- a — mapping from name to IP
- MX— email handling
- NS—name server
- cname
  - subdomains

screenshot from godaddy.com



Type	Name	Value	TTL
A	@	162.244.65.250	600 seconds
CNAME	email	email.fields43.com	1 Hour
CNAME	ftp	@	1 Hour
CNAME	wraith	@	1 Hour
CNAME	www	@	1 Hour
CNAME	_domainconnect	_domainconnect.gd.domaincontrol.com	1 Hour
MX	@	@ (Priority: 0)	1/2 Hour
MX	@	@ (Priority: 10)	1 Hour
NS	@	ns57.domaincontrol.com	1 Hour
NS	@	ns58.domaincontrol.com	1 Hour
SOA	@	Primary nameserver: ns57.domaincontrol.co...	600 seconds

DNS lookup

- To get to powerpuff.cs.brynmawr.edu you could then
  - ask .edu agent how to get to brynmawr.edu
    - then ask brynmawr.edu agent
      - …
- Realistically, almost never happens.
  - DNS servers

DNS lookup tools

- UNIX> dig
  - shows most of the info from godaddy…
- UNIX> nslookup
  - simplest to use
- UNIX> host -t [a DNSname]
  - note the a is for the “a record”

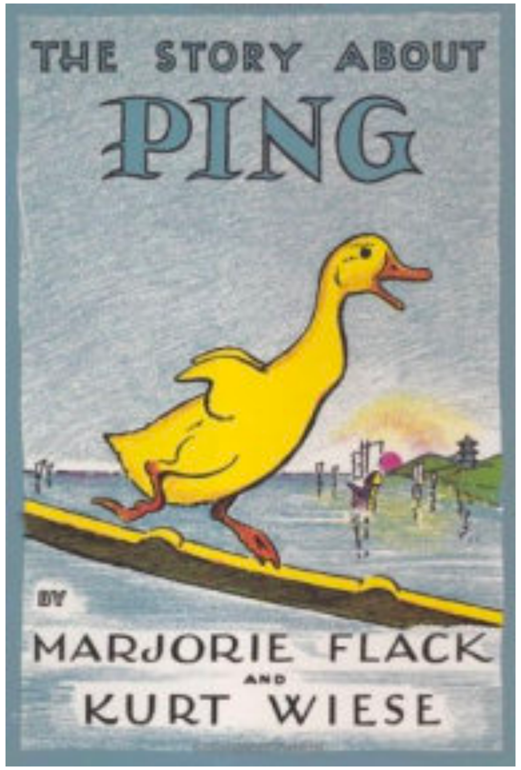
```
[gtowell@powerpuff ~]$ host -t a brynmawr.edu
brynmawr.edu has address 3.225.214.100
brynmawr.edu has address 18.211.178.199

[gtowell@powerpuff ~]$ nslookup brynmawr.edu
Server:         165.106.148.5
Address: 165.106.148.5#53

Name:   brynmawr.edu
Address: 18.211.178.199
Name:   brynmawr.edu
Address: 3.225.214.100

[gtowell@powerpuff ~]$ nslookup powerpuff.cs.brynmawr.e
Server:         165.106.148.5
Address: 165.106.148.5#53

Name:   powerpuff.cs.brynmawr.edu
Address: 165.106.10.113

[gtowell@powerpuff ~]$ hostname -i
165.106.10.113
```

Unix: Network tools

- Ping
  - is machine reachable from your machine and
  - if so, an indication of connection speed and reliability

```
geoffreytowell@Geoff2020Mac Lectures % ping cs.brynmawr.edu
PING cs.brynmawr.edu (165.106.10.107): 56 data bytes
64 bytes from 165.106.10.107: icmp_seq=0 ttl=246 time=21.515 ms
64 bytes from 165.106.10.107: icmp_seq=1 ttl=246 time=33.871 ms
64 bytes from 165.106.10.107: icmp_seq=2 ttl=246 time=26.430 ms
^C
--- cs.brynmawr.edu ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 21.515/27.272/33.871/5.079 ms
geoffreytowell@Geoff2020Mac Lectures % ping powerpuff.cs.brynmawr.ec
PING powerpuff.cs.brynmawr.edu (165.106.10.113): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
^C
--- powerpuff.cs.brynmawr.edu ping statistics ---
4 packets transmitted, 0 packets received, 100.0% packet loss
geoffreytowell@Geoff2020Mac Lectures % ping google.com
PING google.com (142.250.64.110): 56 data bytes
64 bytes from 142.250.64.110: icmp_seq=0 ttl=119 time=18.052 ms
64 bytes from 142.250.64.110: icmp_seq=1 ttl=119 time=29.780 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 2 packets received, 33.3% packet loss
round-trip min/avg/max/stddev = 18.052/23.916/29.780/5.864 ms
```

Ping
Higher level



Using deft allegory, the authors have provided an insightful and intuitive explanation of one of Unix's most venerable networking utilities. Even more stunning is that they were clearly working with a very early beta of the program, as their book first appeared in 1933, years (decades!) before the operating system and network infrastructure were finalized.

The book describes networking in terms even a child could understand, choosing to anthropomorphize the underlying packet structure. The ping packet is described as a duck, who, with other packets (more ducks), spends a certain period of time on the host machine (the wise-eyed boat). At the same time each day (I suspect this is scheduled under cron), the little packets (ducks) exit the host (boat) by way of a bridge (a bridge). From the bridge, the packets travel onto the internet (here embodied by the Yangtze River).

The title character -- er, packet, is called Ping. Ping meanders around the river before being received by another host (another boat). He spends a brief time on the other boat, but eventually returns to his original host machine (the wise-eyed boat) somewhat the worse for wear.

If you need a good, high-level overview of the ping utility, this is the book. I can't recommend it for most managers, as the technical aspects may be too overwhelming and the basic concepts too daunting.

As good as it is, The Story About Ping is not without its faults. There is no index, and though the ping(8) man pages cover the command line options well enough, some review of them seems to be in order. Likewise, in a book solely about Ping, I would have expected a more detailed overview of the ICMP packet structure.

But even with these problems, The Story About Ping has earned a place on my bookshelf, right between Stevens' Advanced Programming in the Unix Environment, and my dog-eared copy of Dante's seminal work on MS Windows, Inferno. Who can read that passage on the Windows API ("Obscure, profound it was, and nebulous, So that by fixing on its depths my sight -- Nothing whatever I discerned therein."), without shaking their head with deep understanding. But I digress.

https://blog.codinghorror.com/the-story-about-ping/

traceroute

- how did my communication get from here to there

```
traceroute to google.ru (172.217.11.35), 30 hops max, 60 byte packets
 1  compsci-gw.brynmawr.edu (165.106.10.1)  1.382 ms  2.746 ms  2.782 ms
 2  gateway-router.brynmawr.edu (165.106.254.26)  0.390 ms  0.373 ms  0.356 ms
 3  204.238.76.61 (204.238.76.61)  1.262 ms  1.245 ms  1.228 ms
 4  204.238.76.235 (204.238.76.235)  1.616 ms  1.599 ms  1.631 ms
 5  lo-0.8.rtsw.newy32aoa.net.internet2.edu (64.57.21.235)  3.864 ms  3.847 ms  3
 6  162.252.69.135 (162.252.69.135)  3.954 ms  3.997 ms  4.056 ms
 7  * 108.170.248.1 (108.170.248.1)  6.078 ms *
 8  108.170.248.33 (108.170.248.33)  4.978 ms 172.253.70.7 (172.253.70.7)  4.318
(108.170.237.206)  4.452 ms
 9  lga25s61-in-f3.1e100.net (172.217.11.35)  4.216 ms  4.301 ms 172.253.69.219 (
[gtowell@benz ~]> traceroute google.com
traceroute to google.com (172.217.10.142), 30 hops max, 60 byte packets
 1  compsci-gw.brynmawr.edu (165.106.10.1)  1.539 ms  1.554 ms  1.535 ms
 2  gateway-router.brynmawr.edu (165.106.254.26)  0.379 ms  0.363 ms  0.563 ms
 3  204.238.76.61 (204.238.76.61)  1.353 ms  1.336 ms  1.318 ms
 4  204.238.76.235 (204.238.76.235)  1.850 ms  1.832 ms  1.869 ms
 5  lo-0.8.rtsw.newy32aoa.net.internet2.edu (64.57.21.235)  4.136 ms  4.119 ms  4
 6  162.252.69.135 (162.252.69.135)  4.085 ms  3.878 ms  3.937 ms
 7  108.170.248.33 (108.170.248.33)  4.785 ms * *
 8  142.250.46.194 (142.250.46.194)  5.465 ms 172.253.71.165 (172.253.71.165)  3
(142.250.235.236)  5.489 ms
 9  172.253.71.165 (172.253.71.165)  3.999 ms 108.170.248.84 (108.170.248.84)  4
(172.253.71.165)  3.908 ms
10  lga34s16-in-f14.1e100.net (172.217.10.142)  3.881 ms  3.900 ms  3.917 ms
```

Bits

- unsigned char val = 0;
    - 00000000
- val = 1;
    - 00000001
- val = 2;
    - 00000010
- val = 4;
    - 00000100
- etc

C (and Java) allow you to work directly on bits

- <<, >>, |, &, ^, ~
    - << and >>
        - NOT related to <, >
        - these “shift”. Multiply / divide by powers of 2
    - |, & are related
        - “bitwise or” and “bitwise and”
    - ^, ~
        - bitwise XOR and bitwise complement

| and &

- bit or
  - 0001001 | 10001000 => 10011001
  - 11110001 | 10101010 => 11111011
  - if there is a 1 in a position in either, then 1 is in the result

- bit and
  - 00010001 & 10001000 =>00000000
  - 11110001 | 10101010 => 10100000
  - if there is a 1 a position in BOTH, then 1 in result
- bit XOR (^) and complement follow

Bit Masks

- A quick way to select store and transfer options. Used a lot, especially in legacy programs

- Idea is to put all selected options into single variable, using bitwise OR
- Then evaluate using a “mask” via bitwise &. Get true if a bit in container is shared with masker

This will require examples

```
//file: masker.c

#define OPTA 1
#define OPTB 2
#define OPTC 4
#define OPTD 8

char getUserSel() {
    char selectedOpts = 0;
    char c[100];
    while (1) {
        printf("Enter a letter: ");
        fgets(c, 99, stdin);
        switch (c[0]) {
            case 'a': case 'A':
                selectedOpts = selectedOpts | OPTA;
                break;
            case 'b': case 'B':
                selectedOpts |= OPTB;
                break;
            case 'c': case 'C':
                selectedOpts |= OPTC;
                break;
            case 'd': case 'D':
                selectedOpts |= OPTD;
                break;
            default:
                return selectedOpts;
            }
    }
}
int main(int argc, char const *argv[])
{
    char userSel = getUserSel();
    printf("%d  %d   %d\n", userSel, (OPTA | OPTB), (userSel & (
    if (userSel & (OPTA | OPTB))
        printf("user selected A or B\n");
    if (userSel & (OPTA | OPTC | OPTD))
        printf("user selected A or C or D\n");
    return 0;
}
```

Bit Shifting

- << n
    - equivalent to multiplying by that power of 2
    - suppose
        - unsigned char x = 16;
            - 00010000
        - x = x << 2;
            - 01000000
            - or x == 64
- >> n
    - equivalent to dividing by that power of n
        - x >>= 3;
            - 00001000
            - or x == 8

```
int main(int argc, char const *argv[])
{
    unsigned char uc = 0;
    uc = 1;
    for (int i = 0; i < 12; i++) {
        printf("Step:%2d  UC:%6d\n", i, uc);
        //uc = uc << 1;
        uc <<= 1;
    }
    uc = 255;
    unsigned char uuc = uc;
    for (int i = 0; i < 10; i++)
    {
        printf("Down:%2d  UC:%6d  UUC:%6d\n",
               i, uc, uuc);
        uc >>= 2;
        uuc /= 4;
    }
}
```

Writing Bits into vars

- Simple math will do it as long as there are 0's
    - unsigned char x = 0;
    - x = 1;
        - 00000001
    - x += 4
        - 00000101 (5)
- But addition falls apart
    - x+=4
        - 00001001 (9)
- Need to use BIT operators
    - x=0;
    - x|=1; //00000001
    - x|=4; //00000101
    - x|=4; //00000101

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char const *argv[])
{
    unsigned char aa = 0;
    for (int i = 0; i < 8; i++)
    {
        if (argv[1][i] == '\0')
            break;
        aa <<= 1;
        if (argv[1][i] == '1')
            aa |= 1;
    }
    printf("%d\n", aa);
    return 0;
}
```

Reading bits from vars

- Same idea, just reversed!
- Basic approach
  - determine if there is a 1 in least significant bit
    - yes: output 1
    - no: output 0
  - divide by 2
  - Repeat

```c
#include <stdio.h>
#include <stdlib.h>
void pbMATH(unsigned char xx, char rtn[]) {
    for (int i = 7; i >=0; i--)
    {
        rtn[i] = '0' + (xx % 2);
        //printf("bit num=%1d  value=%1d\n", i, xx % 2);
        xx /= 2;
    }
}
void pbBITS(int xx, char rtn[]) {
for (int i = 7; i >=0; i--)
    {
        rtn[i] = '0' + (xx&1);
        //printf("bit num=%1d  value=%1d\n", i, xx % 2);
        xx >>= 1;
    }
}
int main(int argc, char const *argv[])
{
    char aa[8];
    for (int i = 1; i < argc; i++)
    {
        pbMATH(atoi(argv[i]), aa);
        printf("BITS:%s\n", aa);
        pbBITS(atoi(argv[i]), aa);
        printf("BITS:%s\n", aa);
    }
    return 0;
}
```

Reading alternate

- Determine if there is a 2 in most significant
  - output 1 or 0
  - multiply by 2
  - repeat

- Usually more convenient to go this way

```
#include <stdio.h>
#include<stdlib.h>

void pbBITSUP(int xx, char rtn[]) {
    unsigned int val = 1 << ((8 * sizeof(int)) - 1);
    for (int i = 0; i < 8 * sizeof(int); i++)
    {
        rtn[i] = '0' + ((xx&val)?1:0);
       xx <<= 1; // shift bits to left
    }
}
int main(int argc, char const *argv[])
{
    char aa[8*sizeof(int)+1];
    aa[8 * sizeof(int) + 1] = '\0';
    aa[8 * sizeof(int)] = '\0';
        for (int i = 1; i < argc; i++)
    {
        pbBITSUP(atoi(argv[i]), aa);
        printf("BITS:%s\n", aa);
    }
    return 0;
}
```

Lab

- Write a program that can take 8, 3 bit numbers and output a string of 24 bits
  - For example:
    - a.out 0 1 2 3 4 5 6 7
    - 000001010011100101110111
  - (read each number as a char, then take only the 3 least significant bits from the char)
- You can just accumulate the bits in a character string or write directly to stdout.

- For an additional complication (not required or even recommended)
  - write the 8 3 bit numbers into 3 8 bit numbers and then output those numbers. This will mean that some 3 bit number are represented across 2 8 bit numbers.