

CS246
Unix:ls -l
C:recursion

March 11

Midterm

- Average: 86, std dev 11
 - Q1: a=15.8 sd=3.0
 - Q2: a=17.5 sd=3.8
 - Q3: a=18.25 sd=2.4
 - Q4: a=18.66 sd=3.7
 - Q5: a=15.33 sd=4.0
 - EC: 9 points total awarded
- full credit answers are posted on the web site.

|s -|

- Reading across
 - char 0: file type: d==directory, l=soft link, -=regular file
 - chars 1-10: permissions
- Column 1 ???
- Column 2: file owner
- Column 3: file group
- Column 4: file size
- Columns 5,6,7: modification date and time
- Column 8: file name

```
[gtowell@powerpuff ~]$ ls -l
total 391500
drwxr-x---  5 gtowell faculty          5 Feb  3 17:17 206
-rw-r--r--  1 gtowell faculty 11633094 Feb  4 21:19 206Public.tgz
drwxr-xr-x  4 gtowell faculty          4 Mar 11 09:24 246
-rw-r--r--  1 gtowell faculty        35 Feb 16 14:46 aaa
drwxrwxrwx  3 gtowell faculty          3 Nov  9 10:30 Android
drwxr-xr-x  3 gtowell faculty          3 Nov  9 10:38 AndroidStudioProjects
drwxr-xr-x  3 gtowell faculty          3 Nov  9 10:28 AStudio
drwxr-xr-x  2 gtowell faculty        21 Feb 23 10:00 bin
```

Owners and groups

- UNIX> whoami
 - shows who you are logged in as
- UNIX> groups
 - shows the groups you are a member of
 - in big installations you might be members of several groups
- When create a file is is created with the current login as owner and the default group as group
 - chown
 - chgrp
 - Not permitted on our systems

Permissions

- permissions are given by 3 triples
 - each triple:
 - READ: r or - reading the file is permitted (or not)
 - WRITE: w or - writing to the file is permitted (or not)
 - EXECUTE: x or - executing the file is permitted (or not)
 - for directories, x means can get a listing
 - letter means permitted, - means not
 - ex: r - -, r-x, or rwx
- First triple == what the file owner is allowed to do
- Second triple = what people in the group are allowed to do
- Third triple = what everyone is allowed to do
- so rwxrw-r- - means that the owner can read the file, write the file and execute, the groups members can read and write and anyone can read

Changing Permissions

- So, for example, to execute a file , you must have the right permission
- As file owner you can change permissions
 - `chmod XXX filename`
 - X is a number in 0-7
 - read = 4 (r=4, -=0)
 - write = 2 (w=2, -=0)
 - execute = 1 (x=1, -=0)
- `chmod 777` = you, the group and everyone can read, write and execute.
- `chmod 774` = you and the group can RWX, everyone else can only read
- `chmod 644` = you can read and write, group and everyone only read
- `chmod 400` = you can read, no one else can do anything

from HW3

- `chmod 777 script`
 - this is plain text file and originally would have been
 - `"rw-r--r--"`
 - So by `chmod 777` you are telling Unix that the file should be treated as runnable. While that, the script is just a text file and will not be run.
- Consider the opposite
 - `gcc xx.c`
 - `chmod 666 a.out`
 - `./a.out`
 - `"permission denied"!`

Recursion

- Used in HW3
 - Not a huge shock, C has it
 - You have used it.
- Max recursion depth
 - Java ~ 10,000
 - C dependent on memory used

```
file: mdr.c
```

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

```
int rec(int d) {  
    if (d>1000000)  
        return 0;  
    fprintf(stderr, "%d\n", d);  
    return rec(d + 1);  
}
```

```
int main(int argc, char const *argv[])  
{  
    rec(1);  
    return 0;  
}
```


Stack Frames

- Things on the call stack are “stack frames”
- frames are “independent” of each other.
 - Communication is limited to passed variables and return values
 - pass by reference
- Recursion limits:
 - Java is based on number of frames
 - C based on total memory used by frames
 - main is itself a stack frame

```
int binarySearch(int arr[], int l, int r, int x, int rep) {
    printf("BSEARCH rep:%d low:%d high:%d\n", rep, l, r);
    if (r >= l) {
        int mid = l + (r - l)/2;
        // If the element is present at the middle itself
        if (arr[mid] == x) {
            void* callstack[128];
            int i, frames = backtrace(callstack, 128);
            char** strs = backtrace_symbols(callstack, frames);
            for (i = 0; i < frames; ++i) {
                printf("%s\n", strs[i]);
            }
            return mid;
        }
        // If element is smaller than mid, then it can only be present
        // in left subarray
        if (arr[mid] > x) return binarySearch(arr, l, mid-1, x, rep);
        return binarySearch(arr, mid+1, r, x, rep+1);
    }
    return -1;
}
```

Recursion and tail recursion

- strcmp in <string.h>
- “tail recursion”
 - the last thing done in the recursion is the recursive call AND nothing is done to the return value

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

```
int strcmprec(char * str1, char * str2) {  
    if (*str1 == '\0') {  
        return *str1 == *str2;  
    }  
    if (*str2 == '\0') {  
        return 0;  
    }  
    if (*str1 != *str2)  
    {  
        return 0;  
    }  
    return strcmprec(++str1, ++str2);  
}
```



Why not str1++?

```
int main(int argc, char const *argv[])  
{  
    printf("%s %s %s\n", argv[1], argv[2], strcmprec(argv[1], argv[2]));  
    printf("DO NOT match");  
    return 0;  
}
```

Tail Recursion

- Writing to avoid tail recursion
 - often just a matter of passing down the value to be returned
- Can always be re-implemented as a loop
 - why bother
 - gcc -O2 will “optimize tail recursion”

file: tr.c

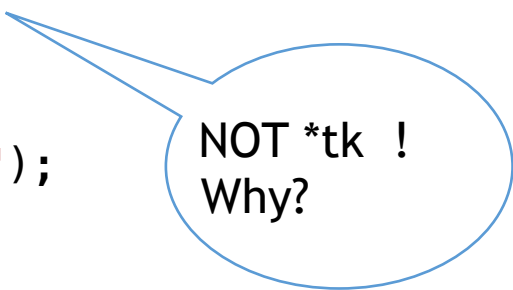
```
int fibTR(int n1, int n2) {  
    int n3 = n1 + n2;  
    if (n3 > 1000)  
        return 1;  
    return 1 + fibTR(n2, n3);  
}
```

```
int fibNTR(int n1, int n2, int i) {  
    int n3 = n1 + n2;  
    if (n3 > 1000)  
        return i;  
    return fibNTR(n2, n3, i + 1);  
}
```

strtok, again

- strtok is useful for parsing well structured files,
- for instance, comma or tab separated columns

```
//12:54 AM 73 F 65 F 76 % S 8 mph 0 mph 3
int main(void)
{
    char aa[512];
    while (NULL != fgets(aa, 1000, stdin))
    {
        fprintf(stderr, "%s\n", aa);
        char *tk = strtok(aa, "\t");
        fprintf(stderr, "Hello\n");
        fprintf(stderr, "%s\n", tk);
        while(1)
        {
            tk = strtok(NULL, "\t");
            if (tk==NULL)
                break;
            fprintf(stderr, "%s\n", tk);
        }
    }
}
```



NOT *tk !
Why?

more with Define

- In addition to simple substitutions can define functions
- WHY
 - define functions are NOT typed.

file: defsqr.c

```
#define SQR(x) (x*x)
```

```
int main(int argc, char const *argv[])  
{  
    double dd = 4.2;  
    int ii = 12;  
    printf("%f\n", SQR(dd));  
    printf("%d\n", SQR(ii));  
    return 0;  
}
```

Lab

- Write a tail recursive and non-tail recursive version of a factorial function
 - $\text{factorial}(3) = 3*2*1 = 6$
 - $\text{factorial}(5) = 5*4*3*2*1 = 120$
- Your factorial functions should return a number of type long