HW 6

Joshua Thompson

27 FEB 2017

Questions

1. (5 points) Why does the operating system perform system calls as oppose to having each user perform the same operations themselves?

   Because the user could do damage to the operating system if they write code at such a low level.

2. (10 points) Look up the following C functions in the man page, label them as either a system call or not a system call.

   (a) fread()

   not a system call

   (b) write()

   system call

   (c) stat()

   system call

   (d) mmap()

   system call

   (e) execv()

   not a system call

3. (10 points) Run `ic221-up`. In the `hw/06/prob3` directory you'll find a compiled program. Use `ltrace` to enumerate the library function calls occurring under `main()`.

   The program uses the strlen, puts, fflush and exit library calls.

4. (10 points) Run `ic221-up`. In the `hw/06/prob4` directory you'll find a compiled program. Use `strace` to enumerate the system function calls occurring under `main()`.

There are several system calls being used. brk, openat, fstat, read, fstat, mmap, mprotect, close, stat, write.

5. (20 points) Consider a file, `accts.dat`, which stores 1000 accounts formatted based on the defined structure. Using `open()` and `read()`, complete the program below to print them out:

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

typedef struct{
  long acctnum;
  double bal;
  char acctname[1024];
} acct_t;

void read_accts(accts){
  //COMPLETE ME
}

int main(int argc, char *argv[]){
  acct_t accts[1000];

  read_accts(accts);

  int i;
  for(i=0;i<1000;i++){
    printf("%ld (%f) -- %s\n",
           accts[i].acctnum,
           accts[i].bal,
           accts[i].acctname);
  }

  close(fd);

}
```

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

typedef struct
{
  long acctnum;
  double bal;
  char acctname[1024];
} acct_t;

void read_accts(acct_t** accts, char* fname)
{
  int fd;
  fd = open(fname, O_RDONLY);
  read(fd, &accts, 1000*sizeof(acct_t));
  close(fd);
}

int main(int argc, char *argv[]) {
  acct_t* accts;
  read_accts(&accts, argv[1]);

  int i;
  for(i = 0; i < 1000; i++)
  {
    printf("%ld (%f) -- %s\n", accts[i].acctnum, accts[i].bal, accts[i].acctname);
  }
  return 0;
}
```

6. (10 points) Complete the following ORing options that matching the `fopen()` permissions:

   (a) `r`

   ```
   O_RONLY
   ```

   (b) `w`

   ```
   O_WRONLY
   ```

   (c) `a`

   ```
   O_APPEND
   ```

   (d) `w+`

   ```
   O_CREATE
   ```

   (e) `r+`

   ```
   O_APPEND
   ```

7. (10 points) Consider a `umask` of 0750 (the leading 0 is to indicate a number written in octal). For the following `open()` permissions, what actual permission will the file get? You can write your answers in octal.

   (a) 0777

   027

   (b) 0640

   020

   (c) 0740

   020

   (d) 0501

001

(e) 0651

021

8. (5 points) Explain why the umask is considered a security feature.

umask ensures that we do not create a file with more permissions than what we want.