332:579 Computer System Project 1

Problem1

Part1:

command: `$gcc -Wall -o myfunc.c myfunc.c`

`gcc` is the compiler for c programming language, it has many flags. Flag `-Wall` will show all the warnings during the file compiling. For the flag of `-o`, it will compile the second argument (a c file) after `-o` into an executable file named same as the first argument after `-o`. So this command will compile c file `myfunc.c` into executable file `myfunc.c`. Then `myfunc.c` will be overwritten with the compilation of this command, which would cause the data loss.

Part2:

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   if (argc < 3 || argc > 4) {
      printf("Usage:./myfiles file_in_1 file_in_2
[file_out (default:myfile.out)]");
   }
   else {
      // Open two files to be merged
      FILE *fp1 = fopen(argv[1], "r");
      FILE *fp2 = fopen(argv[2], "r");
      FILE *fp3;

      char *file_out;
      // Open file to store the result
      if (argc == 3) {
         FILE *fp3 = fopen("myfile.out", "w");
         file_out = "myfile.out";
      }
      else {
         FILE *fp3 = fopen(argv[3], "w");
         file_out = argv[3];
      }
      char c;
```

```
      if (fp1 == NULL || fp2 == NULL || fp3 == NULL) {
         puts("No such file or directory");
         exit(0);
      }

      // Copy contents of first file to file3.txt
      while ((c = fgetc(fp1)) != EOF)
      fputc(c, fp3);

      // Copy contents of second file to file3.txt
      while ((c = fgetc(fp2)) != EOF)
      fputc(c, fp3);
      printf("Merged %s and %s into %s", argv[1], argv[2],
file_out);

      fclose(fp1);
      fclose(fp2);
      fclose(fp3);
      return 0;
   }
}
```

Part3:

```
sz519@engsoft:~/CompSys$ strace -o myfile
usage: strace [-CdffhiqrtttTvVxxy] [-I n] [-e expr]...
              [-a column] [-o file] [-s strsize] [-P
path]...
              -p pid... / [-D] [-E var=val]... [-u
username] PROG [ARGS]
   or: strace -c[df] [-I n] [-e expr]... [-O overhead] [-S
sortby]
              -p pid... / [-D] [-E var=val]... [-u
username] PROG [ARGS]
-c -- count time, calls, and errors for each syscall and
report summary
-C -- like -c but also print regular output
-d -- enable debug output to stderr
-D -- run tracer process as a detached grandchild, not as
parent
-f -- follow forks, -ff -- with output into separate files
-i -- print instruction pointer at time of syscall
-q -- suppress messages about attaching, detaching, etc.
-r -- print relative timestamp, -t -- absolute timestamp,
-tt -- with usecs
-T -- print time spent in each syscall
-v -- verbose mode: print unabbreviated argv, stat,
termios, etc. args
```

```
-x -- print non-ascii strings in hex, -xx -- print all
strings in hex
-y -- print paths associated with file descriptor
arguments
-h -- print help message, -V -- print version
-a column -- alignment COLUMN for printing syscall results
(default 40)
-b execve -- detach on this syscall
-e expr -- a qualifying expression: option=[!]all or
option=[!]val1[,val2]...
   options: trace, abbrev, verbose, raw, signal, read,
write
-I interruptible --
   1: no signals are blocked
   2: fatal signals are blocked while decoding syscall
(default)
   3: fatal signals are always blocked (default if '-o
FILE PROG')
   4: fatal signals and SIGTSTP (^Z) are always blocked
      (useful to make 'strace -o FILE PROG' not stop on
^Z)
-o file -- send trace output to FILE instead of stderr
-O overhead -- set overhead for tracing syscalls to
OVERHEAD usecs
-p pid -- trace process with process id PID, may be
repeated
-s strsize -- limit length of print strings to STRSIZE
chars (default 32)
-S sortby -- sort syscall counts by: time, calls, name,
nothing (default time)
-u username -- run command as username handling setuid
and/or setgid
-E var=val -- put var=val in the environment for command
-E var -- remove var from the environment for command
-P path -- trace accesses to path
```