Doug Branton
COSC519
Homework 2

1.Modify the hello.c program to open an input file (input.txt), read from the input file, and write to another output file (output.txt). This program reads text from one file and writes to another file. Create some text data in the input file and verify that the same data is written to the output file. Understand how a system call is invoked and how it works by generating and reading an ASM file. Identify and mark the system calls in your ASM file. Submit your hello.c and ASM files showing the system calls (Use Linux).

Hello.c:

```
//This is first program
//Dr. Karne
//hello.c
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char **argv)
{
  char c1;
  unsigned char c2;
  int i1=0;
  long l2=0;
  char *cptr;
  int *iptr;
  long *lptr;
  char array1[40] = "This is a string";

  cptr = (char *)malloc(200);
  iptr = (int *)malloc(200);
  lptr = (long *)malloc(200);

  c1 = 'X';
  c2 = 0x44;
  i1 = 0x100;
  l2 = 0x0123456789abcdef;

  *iptr = 0x2000;
  *lptr = 0x88889999aaaabbbb;

  printf("Hello World\n");
  printf("\n\n");
  printf("l2: %lx \n", l2);
  printf("i1: %x \n", i1);
  printf("i1: %10x \n", i1);
  printf("i1: %4x \n", i1);
  printf("c1: %c \n", c1);
```

```
  printf("string: %s \n", array1);

  /*Copy from input.txt to output.txt */
  FILE *input = fopen("input.txt","r");
  FILE *output = fopen("output.txt", "w");

  char ch;

  while((ch = fgetc(input)) != EOF)
    fputc(ch, output);

  fclose(input);
  fclose(output);


  return 0;
}
```

Hello.s (ASM File):

```
.file   "hello.c"
        .intel_syntax noprefix
        .text
.Ltext0:
        .section        .rodata
.LC0:
        .string "Hello World"
.LC1:
        .string "\n"
.LC2:
        .string "l2: %lx \n"
.LC3:
        .string "i1: %x \n"
.LC4:
        .string "i1: %10x \n"
.LC5:
        .string "i1: %4x \n"
.LC6:
        .string "c1: %c \n"
.LC7:
        .string "string: %s \n"
.LC8:
        .string "r"
.LC9:
        .string "input.txt"
.LC10:
        .string "w"
.LC11:
        .string "output.txt"
```

```
        .text
        .globl  main
        .type   main, @function
main:
.LFB6:
        .file 1 "hello.c"
        .loc 1 8 1
        .cfi_startproc
        endbr64
        push    rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        mov     rbp, rsp
        .cfi_def_cfa_register 6
        add     rsp, -128
        mov     DWORD PTR -116[rbp], edi
        mov     QWORD PTR -128[rbp], rsi
        .loc 1 8 1
        mov     rax, QWORD PTR fs:40
        mov     QWORD PTR -8[rbp], rax
        xor     eax, eax
        .loc 1 11 8
        mov     DWORD PTR -100[rbp], 0
        .loc 1 12 9
        mov     QWORD PTR -96[rbp], 0
        .loc 1 16 9
        movabs          rax, 2338328219631577172
        movabs          rdx, 7453010373645639777
        mov     QWORD PTR -48[rbp], rax
        mov     QWORD PTR -40[rbp], rdx
        mov     QWORD PTR -32[rbp], 0
        mov     QWORD PTR -24[rbp], 0
        mov     QWORD PTR -16[rbp], 0
        .loc 1 18 19
        mov     edi, 200
        call    malloc@PLT
        mov     QWORD PTR -88[rbp], rax
        .loc 1 19 18
        mov     edi, 200
        call    malloc@PLT
        mov     QWORD PTR -80[rbp], rax
        .loc 1 20 19
        mov     edi, 200
        call    malloc@PLT
        mov     QWORD PTR -72[rbp], rax
        .loc 1 22 7
        mov     BYTE PTR -103[rbp], 88
        .loc 1 23 7
        mov     BYTE PTR -102[rbp], 68
```

```
.loc 1 24 7
mov     DWORD PTR -100[rbp], 256
.loc 1 25 7
movabs          rax, 81985529216486895
mov     QWORD PTR -96[rbp], rax
.loc 1 27 10
mov     rax, QWORD PTR -80[rbp]
mov     DWORD PTR [rax], 8192
.loc 1 28 10
mov     rax, QWORD PTR -72[rbp]
movabs          rcx, -8608461802446341189
mov     QWORD PTR [rax], rcx
.loc 1 30 4
lea     rdi, .LC0[rip]
call    puts@PLT
.loc 1 31 4
lea     rdi, .LC1[rip]
call    puts@PLT
.loc 1 32 4
mov     rax, QWORD PTR -96[rbp]
mov     rsi, rax
lea     rdi, .LC2[rip]
mov     eax, 0
call    printf@PLT
.loc 1 33 4
mov     eax, DWORD PTR -100[rbp]
mov     esi, eax
lea     rdi, .LC3[rip]
mov     eax, 0
call    printf@PLT
.loc 1 34 4
mov     eax, DWORD PTR -100[rbp]
mov     esi, eax
lea     rdi, .LC4[rip]
mov     eax, 0
call    printf@PLT
.loc 1 35 4
mov     eax, DWORD PTR -100[rbp]
mov     esi, eax
lea     rdi, .LC5[rip]
mov     eax, 0
call    printf@PLT
.loc 1 36 4
movsx eax, BYTE PTR -103[rbp]
mov     esi, eax
lea     rdi, .LC6[rip]
mov     eax, 0
call    printf@PLT
.loc 1 37 4
```

```
        lea     rax, -48[rbp]
        mov     rsi, rax
        lea     rdi, .LC7[rip]
        mov     eax, 0
        call    printf@PLT
        .loc 1 40 18
        lea     rsi, .LC8[rip]
        lea     rdi, .LC9[rip]
        call    fopen@PLT
        mov     QWORD PTR -64[rbp], rax
        .loc 1 41 19
        lea     rsi, .LC10[rip]
        lea     rdi, .LC11[rip]
        call    fopen@PLT
        mov     QWORD PTR -56[rbp], rax
        .loc 1 45 9
        jmp     .L2
.L3:
        .loc 1 46 7
        movsx eax, BYTE PTR -101[rbp]
        mov     rdx, QWORD PTR -56[rbp]
        mov     rsi, rdx
        mov     edi, eax
        call    fputc@PLT
.L2:
        .loc 1 45 16
        mov     rax, QWORD PTR -64[rbp]
        mov     rdi, rax
        call    fgetc@PLT
        .loc 1 45 14
        mov     BYTE PTR -101[rbp], al
        .loc 1 45 9
        cmp     BYTE PTR -101[rbp], -1
        jne     .L3
        .loc 1 48 4
        mov     rax, QWORD PTR -64[rbp]
        mov     rdi, rax
        call    fclose@PLT
        .loc 1 49 4
        mov     rax, QWORD PTR -56[rbp]
        mov     rdi, rax
        call    fclose@PLT
        .loc 1 52 11
        mov     eax, 0
        .loc 1 53 1
        mov     rcx, QWORD PTR -8[rbp]
        xor     rcx, QWORD PTR fs:40
        je      .L5
        call    __stack_chk_fail@PLT
```

```
.L5:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE6:
        .size   main, .-main
.Letext0:
        .file 2 "/usr/lib/gcc/x86_64-linux-gnu/9/include/stddef.h"
        .file 3 "/usr/include/x86_64-linux-gnu/bits/types.h"
        .file 4 "/usr/include/x86_64-linux-gnu/bits/types/struct_FILE.h"
        .file 5 "/usr/include/x86_64-linux-gnu/bits/types/FILE.h"
        .file 6 "/usr/include/stdio.h"
        .file 7 "/usr/include/x86_64-linux-gnu/bits/sys_errlist.h"
        .section        .debug_info,"",@progbits
.Ldebug_info0:
```

2. Using the above hello.exe or hello.o files, run objdump command to find system calls and mark them in a file. System calls have UND symbols.