*1. Enhance the hello.c program to open a file, read from the file, write to the file, and close the 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).*

**Source Code – hello.c (built using GCC)**

/\* Hello World program \*/

#include<stdio.h>

#include<stdlib.h>

int main()

{

char \*outputFilename = "output.txt";

char ch;

FILE \*ifp, \*ofp;

printf("Hello World");

ifp = fopen("input.txt", "a");

if (ifp == NULL){

fprintf(stderr, "Can't open input file input.txt!\n");

exit(1);

}

ofp = fopen(outputFilename, "a");

if (ofp == NULL) {

fprintf(stderr, "Can't open output file %s!\n",

outputFilename);

exit(1);

}

while (1) {

ch = fgetc(ifp);

if (ch == EOF)

break;

else

putc(ch, ofp);

}

fprintf(ifp, "..appending text to INPUT file.");

fprintf(ofp, "..appending text to OUTPUT file.");

fclose(ifp);

fclose(ofp);

return 0;

}

\* Note: Systems calls are marked by the **bold** **red** text.

**GCC/Linux Generated hello.s Assembly File**

.file "hello.c"

.section .rodata

.LC0:

.string "output.txt"

.LC1:

.string "Hello World"

.LC2:

.string "a"

.LC3:

.string "input.txt"

.align 8

.LC4:

.string "Can't open input file input.txt!\n"

.LC5:

.string "Can't open output file %s!\n"

.align 8

.LC6:

.string "..appending text to INPUT file."

.align 8

.LC7:

.string "..appending text to OUTPUT file."

.text

.globl main

.type main, @function

main:

.LFB2:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $32, %rsp

movq $.LC0, -8(%rbp)

movl $.LC1, %edi

movl $0, %eax

**call printf**

movl $.LC2, %esi

movl $.LC3, %edi

**call fopen**

movq %rax, -16(%rbp)

cmpq $0, -16(%rbp)

jne .L2

movq stderr(%rip), %rax

movq %rax, %rcx

movl $33, %edx

movl $1, %esi

movl $.LC4, %edi

**call fwrite**

movl $1, %edi

**call exit**

.L2:

movq -8(%rbp), %rax

movl $.LC2, %esi

movq %rax, %rdi

**call fopen**

movq %rax, -24(%rbp)

cmpq $0, -24(%rbp)

jne .L3

movq stderr(%rip), %rax

movq -8(%rbp), %rdx

movl $.LC5, %esi

movq %rax, %rdi

movl $0, %eax

**call fprintf**

movl $1, %edi

**call exit**

.L3:

movq -16(%rbp), %rax

movq %rax, %rdi

**call fgetc**

movb %al, -25(%rbp)

cmpb $-1, -25(%rbp)

jne .L4

jmp .L7

.L4:

movsbl -25(%rbp), %eax

movq -24(%rbp), %rdx

movq %rdx, %rsi

movl %eax, %edi

**call \_IO\_putc**

jmp .L3

.L7:

movq -16(%rbp), %rax

movq %rax, %rcx

movl $31, %edx

movl $1, %esi

movl $.LC6, %edi

**call fwrite**

movq -24(%rbp), %rax

movq %rax, %rcx

movl $32, %edx

movl $1, %esi

movl $.LC7, %edi

**call fwrite**

movq -16(%rbp), %rax

movq %rax, %rdi

**call fclose**

movq -24(%rbp), %rax

movq %rax, %rdi

**call fclose**

movl $0, %eax

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE2:

.size main, .-main

.ident "GCC: (GNU) 4.8.3 20140911 (Red Hat 4.8.3-9)"

.section .note.GNU-stack,"",@progbits

*2. Use the above hello.exe file and objdump command to create an asm file in Linux and mark all system calls in this program. Notice that some are system calls and some are local calls in the asm file. System calls have UND symbols.*

**OBJDUMP Generated hello.s Assembly File**

Script started on Thu 01 Oct 2015 01:26:22 PM EDT

\_]0;user@localhost:~/git/Operating-Systems/Homeworks/Homework2/Homework2Linux/Debug/src\_\_]7;file://localhost.localdomain/home/user/git/Operating-Systems/Homeworks/Homework2/Homework2Linux/Debug/src\_\_[?1034h[user@localhost src]$ d\_\_[Kobjdump -d -t hello.o

hello.o: file format elf64-x86-64

SYMBOL TABLE:

0000000000000000 l df \*ABS\* 0000000000000000 hello.c

0000000000000000 l d .text 0000000000000000 .text

0000000000000000 l d .data 0000000000000000 .data

0000000000000000 l d .bss 0000000000000000 .bss

0000000000000000 l d .rodata 0000000000000000 .rodata

0000000000000000 l d .debug\_info 0000000000000000 .debug\_info

0000000000000000 l d .debug\_abbrev 0000000000000000 .debug\_abbrev

0000000000000000 l d .debug\_aranges 0000000000000000 .debug\_aranges

0000000000000000 l d .debug\_line 0000000000000000 .debug\_line

0000000000000000 l d .debug\_str 0000000000000000 .debug\_str

0000000000000000 l d .note.GNU-stack 0000000000000000 .note.GNU-stack

0000000000000000 l d .eh\_frame 0000000000000000 .eh\_frame

0000000000000000 l d .comment 0000000000000000 .comment

0000000000000000 g F .text 000000000000013d main

0000000000000000 \*UND\* 0000000000000000 \_GLOBAL\_OFFSET\_TABLE\_

0000000000000000 **\*UND\* 0000000000000000 printf**

0000000000000000 **\*UND\* 0000000000000000 fopen**

0000000000000000 **\*UND\* 0000000000000000 stderr**

0000000000000000 **\*UND\* 0000000000000000 fwrite**

0000000000000000 **\*UND\* 0000000000000000 exit**

0000000000000000 **\*UND\* 0000000000000000 fprintf**

0000000000000000 **\*UND\* 0000000000000000 fgetc**

0000000000000000 **\*UND\* 0000000000000000 \_IO\_putc**

0000000000000000 **\*UND\* 0000000000000000 fclose**

Disassembly of section .text:

0000000000000000 <main>:

0: 55 push %rbp

1: 48 89 e5 mov %rsp,%rbp

4: 48 83 ec 20 sub $0x20,%rsp

8: 48 8d 05 00 00 00 00 lea 0x0(%rip),%rax # f <main+0xf>

f: 48 89 45 f8 mov %rax,-0x8(%rbp)

13: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 1a <main+0x1a>

1a: b8 00 00 00 00 mov $0x0,%eax

1f: e8 00 00 00 00 **callq 24 <main+0x24>**

24: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2b <main+0x2b>

2b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 32 <main+0x32>

32: e8 00 00 00 00 **callq 37 <main+0x37>**

37: 48 89 45 f0 mov %rax,-0x10(%rbp)

3b: 48 83 7d f0 00 cmpq $0x0,-0x10(%rbp)

40: 75 2d jne 6f <main+0x6f>

42: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 49 <main+0x49>

49: 48 8b 00 mov (%rax),%rax

4c: 48 89 c1 mov %rax,%rcx

4f: ba 21 00 00 00 mov $0x21,%edx

54: be 01 00 00 00 mov $0x1,%esi

59: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 60 <main+0x60>

60: e8 00 00 00 00 **callq 65 <main+0x65>**

65: bf 01 00 00 00 mov $0x1,%edi

6a: e8 00 00 00 00 **callq 6f <main+0x6f>**

6f: 48 8b 45 f8 mov -0x8(%rbp),%rax

73: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 7a <main+0x7a>

7a: 48 89 c7 mov %rax,%rdi

7d: e8 00 00 00 00 **callq 82 <main+0x82>**

82: 48 89 45 e8 mov %rax,-0x18(%rbp)

86: 48 83 7d e8 00 cmpq $0x0,-0x18(%rbp)

8b: 75 2c jne b9 <main+0xb9>

8d: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 94 <main+0x94>

94: 48 8b 00 mov (%rax),%rax

97: 48 8b 55 f8 mov -0x8(%rbp),%rdx

9b: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # a2 <main+0xa2>

a2: 48 89 c7 mov %rax,%rdi

a5: b8 00 00 00 00 mov $0x0,%eax

aa: e8 00 00 00 00 **callq af <main+0xaf>**

af: bf 01 00 00 00 mov $0x1,%edi

b4: e8 00 00 00 00 **callq b9 <main+0xb9>**

b9: 48 8b 45 f0 mov -0x10(%rbp),%rax

bd: 48 89 c7 mov %rax,%rdi

c0: e8 00 00 00 00 **callq c5 <main+0xc5>**

c5: 88 45 e7 mov %al,-0x19(%rbp)

c8: 80 7d e7 ff cmpb $0xff,-0x19(%rbp)

cc: 75 02 jne d0 <main+0xd0>

ce: eb 14 jmp e4 <main+0xe4>

d0: 0f be 45 e7 movsbl -0x19(%rbp),%eax

d4: 48 8b 55 e8 mov -0x18(%rbp),%rdx

d8: 48 89 d6 mov %rdx,%rsi

db: 89 c7 mov %eax,%edi

dd: e8 00 00 00 00 **callq e2 <main+0xe2>**

e2: eb d5 jmp b9 <main+0xb9>

e4: 48 8b 45 f0 mov -0x10(%rbp),%rax

e8: 48 89 c1 mov %rax,%rcx

eb: ba 1f 00 00 00 mov $0x1f,%edx

f0: be 01 00 00 00 mov $0x1,%esi

f5: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # fc <main+0xfc>

fc: e8 00 00 00 00 **callq 101 <main+0x101>**

101: 48 8b 45 e8 mov -0x18(%rbp),%rax

105: 48 89 c1 mov %rax,%rcx

108: ba 20 00 00 00 mov $0x20,%edx

10d: be 01 00 00 00 mov $0x1,%esi

112: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 119 <main+0x119>

119: e8 00 00 00 00 **callq 11e <main+0x11e>**

11e: 48 8b 45 f0 mov -0x10(%rbp),%rax

122: 48 89 c7 mov %rax,%rdi

125: e8 00 00 00 00 **callq 12a <main+0x12a>**

12a: 48 8b 45 e8 mov -0x18(%rbp),%rax

12e: 48 89 c7 mov %rax,%rdi

131: e8 00 00 00 00 **callq 136 <main+0x136>**

136: b8 00 00 00 00 mov $0x0,%eax

13b: c9 leaveq

13c: c3 retq

\_]0;user@localhost:~/git/Operating-Systems/Homeworks/Homework2/Homework2Linux/Debug/src\_\_]7;file://localhost.localdomain/home/user/git/Operating-Systems/Homeworks/Homework2/Homework2Linux/Debug/src\_[user@localhost src]$ sc\_\_[K\_\_[Kexit

exit

Script done on Thu 01 Oct 2015 01:26:34 PM EDT

*3. Use at least one Windows API call in your program and run it in the Visual Studio environment. Submit your program and output. What is the difference between system call and API?*