**COSC 519 - OS**

**Spring 2014**

**Emanuel Rivera**

**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.**

Commands Ran:

gcc -Wall -S -c "hello.c"

gcc -Wall -o "hello" "hello.c"

**C code is called hello.c**

===============

#include <stdio.h>

int main()

{

char \*inname = "file.txt";

char \*outname = "fileout.txt";

FILE \*infile;

FILE \*outfile;

char line\_buffer[BUFSIZ]; /\* BUFSIZ is defined if you include stdio.h \*/

char line\_number;

infile = fopen(inname, "r");

outfile = fopen(outname,"w");

if (!infile) {

printf("Couldn't open file %s for reading.\n", inname);

return 0;

}

printf("Opened file %s for reading.\n", inname);

line\_number = 0;

while (fgets(line\_buffer, sizeof(line\_buffer), infile)) {

++line\_number;

/\* note that the newline is in the buffer \*/

printf("%4d: %s", line\_number, line\_buffer);

fprintf(outfile,"%4d: %s",line\_number, line\_buffer);

}

printf("\nTotal number of lines = %d\n", line\_number);

fprintf(outfile,"\nTotal number of lines = %d\n", line\_number);

fclose(infile);

fclose(outfile);

return 0;

}

===============

**ASM file is called hello.s**

===================================

.file "hello.c"

.section .rodata

.LC0:

.string "file.txt"

.LC1:

.string "fileout.txt"

.LC2:

.string "r"

.LC3:

.string "w"

.align 8

.LC4:

.string "Couldn't open file %s for reading.\n"

.LC5:

.string "Opened file %s for reading.\n"

.LC6:

.string "%4d: %s"

.LC7:

.string "\nTotal number of lines = %d\n"

.text

.globl main

.type main, @function

main:

.LFB0:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $8256, %rsp

movq %fs:40, %rax

movq %rax, -8(%rbp)

xorl %eax, %eax

movq $.LC0, -8248(%rbp)

movq $.LC1, -8240(%rbp)

movl $.LC2, %edx

movq -8248(%rbp), %rax

movq %rdx, %rsi

movq %rax, %rdi

call fopen

movq %rax, -8232(%rbp)

movl $.LC3, %edx

movq -8240(%rbp), %rax

movq %rdx, %rsi

movq %rax, %rdi

call fopen

movq %rax, -8224(%rbp)

cmpq $0, -8232(%rbp)

jne .L2

movl $.LC4, %eax

movq -8248(%rbp), %rdx

movq %rdx, %rsi

movq %rax, %rdi

movl $0, %eax

call printf

movl $0, %eax

jmp .L3

.L2:

movl $.LC5, %eax

movq -8248(%rbp), %rdx

movq %rdx, %rsi

movq %rax, %rdi

movl $0, %eax

call printf

movb $0, -8209(%rbp)

jmp .L4

.L5:

addb $1, -8209(%rbp)

movsbl -8209(%rbp), %ecx

movl $.LC6, %eax

leaq -8208(%rbp), %rdx

movl %ecx, %esi

movq %rax, %rdi

movl $0, %eax

call printf

movsbl -8209(%rbp), %edx

movl $.LC6, %esi

leaq -8208(%rbp), %rcx

movq -8224(%rbp), %rax

movq %rax, %rdi

movl $0, %eax

call fprintf

.L4:

movq -8232(%rbp), %rdx

leaq -8208(%rbp), %rax

movl $8192, %esi

movq %rax, %rdi

call fgets

testq %rax, %rax

jne .L5

movsbl -8209(%rbp), %edx

movl $.LC7, %eax

movl %edx, %esi

movq %rax, %rdi

movl $0, %eax

call printf

movsbl -8209(%rbp), %edx

movl $.LC7, %ecx

movq -8224(%rbp), %rax

movq %rcx, %rsi

movq %rax, %rdi

movl $0, %eax

call fprintf

movq -8232(%rbp), %rax

movq %rax, %rdi

call fclose

movq -8224(%rbp), %rax

movq %rax, %rdi

call fclose

movl $0, %eax

.L3:

movq -8(%rbp), %rdx

xorq %fs:40, %rdx

je .L6

call \_\_stack\_chk\_fail

.L6:

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE0:

.size main, .-main

.ident "GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3"

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

===================================

**System Calls:**

Line 41: call fopen

Line 47: call fopen

Line 89: call fgets

Line 104: call fprintf

Line 107: call fclose

Line 110: call fclose

**2) Create and run a hello program in Linux. Use 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. You may have to generate an assembly list file to help you to do this work.**

Commands Ran:

objdump -d hello > hello.objdump.asm

**File for Objdump assembly code is called hello.objdump.asm**

System calls:

4006c6: e8 c5 fe ff ff callq 400590 <fopen@plt>

4006e4: e8 a7 fe ff ff callq 400590 <fopen@plt>

400790: e8 eb fd ff ff callq 400580 <fprintf@plt>

4007ab: e8 c0 fd ff ff callq 400570 <fgets@plt>

4007cb: e8 80 fd ff ff callq 400550 <printf@plt>

4007ee: e8 8d fd ff ff callq 400580 <fprintf@plt>

4007fd: e8 2e fd ff ff callq 400530 <fclose@plt>

40080c: e8 1f fd ff ff callq 400530 <fclose@plt>

**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?**

Windows API lets you development a windows appication that can run with no compatibity issues and has the advatage of using features and capabilities unique to each version of windows. A System call is a request to the kernel which is inteded to be very low level interface to the kernel. Windows API are used to invoke system calls.

**APPLICATION:**

==================================================

#include "stdafx.h"

#include <windows.h>

#include <tchar.h>

#include <stdio.h>

#include <iostream>

using namespace std;

#define BUFFER\_SIZE 1024

#define COPY\_SIZE 512

/\*

MyCopyMemory - A wrapper for CopyMemory

buf - destination buffer

pbData - source buffer

cbData - size of block to copy, in bytes

bufsize - size of the destination buffer

\*/

void MyCopyMemory(TCHAR \*buf, TCHAR \*pbData, SIZE\_T cbData, SIZE\_T bufsize)

{

CopyMemory(buf, pbData, min(cbData,bufsize));

}

int main()

{

TCHAR buf[BUFFER\_SIZE] = TEXT("This is the destination");

TCHAR pbData[BUFFER\_SIZE] = TEXT("This is the source");

MyCopyMemory(buf, pbData, COPY\_SIZE\*sizeof(TCHAR), BUFFER\_SIZE\*sizeof(TCHAR));

\_tprintf(TEXT("Destination buffer contents: %s\n"), buf);

int i;

cout << "Pause: ";

cin >> i;

return 0;

}

====================================================

**Output:**

==========================

Destination buffer contents: This is the source

==========================