Kevin Kuo
Homework #2

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 – Linux/GCC*/

#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?

In Windows, you're not supposed to use manual system calls. You utilize a NTDLL and a Native API (such as Win32) to accomplish system calls. The Native API is a wrapper around the kernel mode side. It performs the system call for the correct API.

In the case of Linux, you can perform manual system calls in assembly unlike Windows where you need to work through an extra layer of abstraction (NTDLL and Win32).

Source Code – hello.c (built using MSVS2012)

```
/* Hello World program - Windows MSVS2012 */

#include<stdio.h>
#include<stdlib.h>

main()
{

      char *outputFilename = "output.txt";
      char ch;
```

```
    FILE *ifp, *ofp;

     printf("Hello World");

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

     if (ifp == NULL){
          fprintf(stderr, "Can't open input file in.list!\n");
          exit(1);
     }

     ofp = fopen(outputFilename, "w");

     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);
     }

     fclose(ifp);
     fclose(ofp);
}
```

MSVS2012 Generated hello.asm Assembly File

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 17.00.61219.0

      TITLE C:\Users\Kevin Kuo\git\Operating-
Systems\Homeworks\Homework2\Homework2Windows\Homework2Windows\hello.c
      .686P
      .XMM
      include listing.inc
      .model      flat

INCLUDELIB MSVCRTD
INCLUDELIB OLDNAMES

PUBLIC      _main
PUBLIC      ??_C@_0L@ODNFPCJH@output?4txt?$AA@        ; `string'
PUBLIC      ??_C@_0M@KPLPPDAC@Hello?5World?$AA@       ; `string'
PUBLIC      ??_C@_01KDCPPGHE@r?$AA@                   ; `string'
PUBLIC      ??_C@_09KMIIOAHK@input?4txt?$AA@          ; `string'
PUBLIC      ??_C@_0CA@PIKJCKAP@Can?8t?5open?5input?5file?5in?4list?$CB?6?$AA@ ; `string'
PUBLIC      ??_C@_01NOFIACDB@w?$AA@                   ; `string'
PUBLIC      ??_C@_0BM@FFNLJCMO@Can?8t?5open?5output?5file?5?$CFs?$CB?6?$AA@ ; `string'
```

```
EXTRN __imp____iob_func:PROC
EXTRN __imp__fclose:PROC
EXTRN __imp__fgetc:PROC
EXTRN __imp__fopen:PROC
EXTRN __imp__fprintf:PROC
EXTRN __imp__printf:PROC
EXTRN __imp__putc:PROC
EXTRN __imp__exit:PROC
EXTRN __RTC_CheckEsp:PROC
EXTRN __RTC_InitBase:PROC
EXTRN __RTC_Shutdown:PROC
;     COMDAT rtc$TMZ
rtc$TMZ     SEGMENT
__RTC_Shutdown.rtc$TMZ DD FLAT:__RTC_Shutdown
rtc$TMZ     ENDS
;     COMDAT rtc$IMZ
rtc$IMZ     SEGMENT
__RTC_InitBase.rtc$IMZ DD FLAT:__RTC_InitBase
rtc$IMZ     ENDS
;     COMDAT ??_C@_0BM@FFNLJCMO@Can?8t?5open?5output?5file?5?$CFs?$CB?6?$AA@
CONST SEGMENT
??_C@_0BM@FFNLJCMO@Can?8t?5open?5output?5file?5?$CFs?$CB?6?$AA@ DB 'Can'''
      DB    't open output file %s!', 0aH, 00H        ; `string'
CONST ENDS
;     COMDAT ??_C@_01NOFIACDB@w?$AA@
CONST SEGMENT
??_C@_01NOFIACDB@w?$AA@ DB 'w', 00H             ; `string'
CONST ENDS
;     COMDAT ??_C@_0CA@PIKJCKAP@Can?8t?5open?5input?5file?5in?4list?$CB?6?$AA@
CONST SEGMENT
??_C@_0CA@PIKJCKAP@Can?8t?5open?5input?5file?5in?4list?$CB?6?$AA@ DB 'Can'
      DB    '''t open input file in.list!', 0aH, 00H  ; `string'
CONST ENDS
;     COMDAT ??_C@_09KMIIOAHK@input?4txt?$AA@
CONST SEGMENT
??_C@_09KMIIOAHK@input?4txt?$AA@ DB 'input.txt', 00H  ; `string'
CONST ENDS
;     COMDAT ??_C@_01KDCPPGHE@r?$AA@
CONST SEGMENT
??_C@_01KDCPPGHE@r?$AA@ DB 'r', 00H             ; `string'
CONST ENDS
;     COMDAT ??_C@_0M@KPLPPDAC@Hello?5World?$AA@
CONST SEGMENT
??_C@_0M@KPLPPDAC@Hello?5World?$AA@ DB 'Hello World', 00H ; `string'
CONST ENDS
;     COMDAT ??_C@_0L@ODNFPCJH@output?4txt?$AA@
CONST SEGMENT
??_C@_0L@ODNFPCJH@output?4txt?$AA@ DB 'output.txt', 00H     ; `string'
CONST ENDS
; Function compile flags: /Odtp /RTCsu /ZI
; File c:\users\kevin kuo\git\operating-
systems\homeworks\homework2\homework2windows\homework2windows\hello.c
;     COMDAT _main
_TEXT SEGMENT
_ofp$ = -44                             ; size = 4
_ifp$ = -32                             ; size = 4
_ch$ = -17                              ; size = 1
_outputFilename$ = -8                         ; size = 4
```

```
_main PROC                                      ; COMDAT

; 7    : {

       push  ebp
       mov   ebp, esp
       sub   esp, 240                           ; 000000f0H
       push  ebx
       push  esi
       push  edi
       lea   edi, DWORD PTR [ebp-240]
       mov   ecx, 60                                    ; 0000003cH
       mov   eax, -858993460                            ; ccccccccH
       rep stosd

; 8    :
; 9    :     char *outputFilename = "output.txt";

       mov   DWORD PTR _outputFilename$[ebp], OFFSET ??_C@_0L@ODNFPCJH@output?4txt?$AA@

; 10   :     char ch;
; 11   :
; 12   :      FILE *ifp, *ofp;
; 13   :
; 14   :     printf("Hello World");

       mov   esi, esp
       push  OFFSET ??_C@_0M@KPLPPDAC@Hello?5World?$AA@
       call  DWORD PTR __imp__printf
       add   esp, 4
       cmp   esi, esp
       call  __RTC_CheckEsp

; 15   :
; 16   :     ifp = fopen("input.txt", "r");

       mov   esi, esp
       push  OFFSET ??_C@_01KDCPPGHE@r?$AA@
       push  OFFSET ??_C@_09KMIIOAHK@input?4txt?$AA@
       call  DWORD PTR __imp__fopen
       add   esp, 8
       cmp   esi, esp
       call  __RTC_CheckEsp
       mov   DWORD PTR _ifp$[ebp], eax

; 17   :
; 18   :     if (ifp == NULL){

       cmp   DWORD PTR _ifp$[ebp], 0
       jne   SHORT $LN6@main

; 19   :           fprintf(stderr, "Can't open input file in.list!\n");

       mov   esi, esp
       push  OFFSET ??_C@_0CA@PIKJCKAP@Can?8t?5open?5input?5file?5in?4list?$CB?6?$AA@
       mov   edi, esp
       call  DWORD PTR __imp____iob_func
       cmp   edi, esp
```

```
        call    __RTC_CheckEsp
        mov     ecx, 32                         ; 00000020H
        shl     ecx, 1
        add     eax, ecx
        push    eax
        call    DWORD PTR __imp__fprintf
        add     esp, 8
        cmp     esi, esp
        call    __RTC_CheckEsp

; 20    :           exit(1);

        mov     esi, esp
        push    1
        call    DWORD PTR __imp__exit
        cmp     esi, esp
        call    __RTC_CheckEsp
$LN6@main:

; 21    :     }
; 22    :
; 23    :     ofp = fopen(outputFilename, "w");

        mov     esi, esp
        push    OFFSET ??_C@_01NOFIACDB@w?$AA@
        mov     eax, DWORD PTR _outputFilename$[ebp]
        push    eax
        call    DWORD PTR __imp__fopen
        add     esp, 8
        cmp     esi, esp
        call    __RTC_CheckEsp
        mov     DWORD PTR _ofp$[ebp], eax

; 24    :
; 25    :     if (ofp == NULL) {

        cmp     DWORD PTR _ofp$[ebp], 0
        jne     SHORT $LN4@main

; 26    :           fprintf(stderr, "Can't open output file %s!\n",
; 27    :         outputFilename);

        mov     esi, esp
        mov     eax, DWORD PTR _outputFilename$[ebp]
        push    eax
        push    OFFSET ??_C@_0BM@FFNLJCMO@Can?8t?5open?5output?5file?5?$CFs?$CB?6?$AA@
        mov     edi, esp
        call    DWORD PTR __imp____iob_func
        cmp     edi, esp
        call    __RTC_CheckEsp
        mov     ecx, 32                         ; 00000020H
        shl     ecx, 1
        add     eax, ecx
        push    eax
        call    DWORD PTR __imp__fprintf
        add     esp, 12                         ; 0000000cH
        cmp     esi, esp
        call    __RTC_CheckEsp
```

```
; 28   :          exit(1);

      mov   esi, esp
      push  1
      call  DWORD PTR __imp__exit
      cmp   esi, esp
      call  __RTC_CheckEsp
$LN4@main:

; 29   :    }
; 30   :
; 31   :    while (1) {

      mov   eax, 1
      test  eax, eax
      je    SHORT $LN3@main

; 32   :       ch = fgetc(ifp);

      mov   esi, esp
      mov   eax, DWORD PTR _ifp$[ebp]
      push  eax
      call  DWORD PTR __imp__fgetc
      add   esp, 4
      cmp   esi, esp
      call  __RTC_CheckEsp
      mov   BYTE PTR _ch$[ebp], al

; 33   :
; 34   :       if (ch == EOF)

      movsx eax, BYTE PTR _ch$[ebp]
      cmp   eax, -1
      jne   SHORT $LN2@main

; 35   :          break;

      jmp   SHORT $LN3@main

; 36   :       else

      jmp   SHORT $LN1@main
$LN2@main:

; 37   :          putc(ch, ofp);

      mov   esi, esp
      mov   eax, DWORD PTR _ofp$[ebp]
      push  eax
      movsx ecx, BYTE PTR _ch$[ebp]
      push  ecx
      call  DWORD PTR __imp__putc
      add   esp, 8
      cmp   esi, esp
      call  __RTC_CheckEsp
$LN1@main:
```

```
; 38   :    }

      jmp   SHORT $LN4@main
$LN3@main:

; 39   :
; 40   :    fclose(ifp);

      mov   esi, esp
      mov   eax, DWORD PTR _ifp$[ebp]
      push  eax
      call  DWORD PTR __imp__fclose
      add   esp, 4
      cmp   esi, esp
      call  __RTC_CheckEsp

; 41   :    fclose(ofp);

      mov   esi, esp
      mov   eax, DWORD PTR _ofp$[ebp]
      push  eax
      call  DWORD PTR __imp__fclose
      add   esp, 4
      cmp   esi, esp
      call  __RTC_CheckEsp

; 42   : }

      jmp   SHORT $LN10@main
$LN8@main:
      jmp   SHORT $LN9@main
$LN10@main:
      xor   eax, eax
$LN9@main:
      pop   edi
      pop   esi
      pop   ebx
      add   esp, 240                        ; 000000f0H
      cmp   ebp, esp
      call  __RTC_CheckEsp
      mov   esp, ebp
      pop   ebp
      ret   0
_main ENDP
_TEXT ENDS
END
```

Notes:

- The DWORD ptr is a size directive. It specifies the size of the target operand.