HW#2
Advanced Operating Systems, Spring 2023

MengXian,Du (CBB108047)
Department of Computer Science and Information Engineering
National Pingtung University

1. You have to write C programs for measuring the costs of any 3 system calls and the context switch.

   Solution: Please refer to List 1 (hw2.c):

Listing 1: hw2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <stdint.h>
#include <unistd.h>
#include <sys/wait.h>

uint64_t rdtsc()
{
    uint32_t lo, hi;
    __asm__ __volatile__("rdtsc\n\t"
                         : "=a"(lo), "=d"(hi));
    // low32-bit store in EAX register high32-bit store in EDX register
    /* This code in asm like following
    rdtsc ; read time-stamp count er into edx:eax
    mov lo, eax ; store low 32 bits into lo
    mov hi, edx ; store high 32 bits into hi
    */
    return ((uint64_t)hi << 32) | lo;
}
uint64_t m_start(int s)
{
    printf("----------Measure Start%d------------\n", s);
    uint64_t start = rdtsc();
    return start;
}
double m_end(uint64_t start)
{
    uint64_t end, cycle;
    double sec, nanoSec;
    end = rdtsc();
    cycle = end - start; // calcuate CPU cycle
    sec = (double)cycle / (double)210000000;
    nanoSec = (double)cycle / (double)2.1;
    printf("Cycle : %lu \n", cycle);
    printf("Second: %lfs\n", sec);
    printf("Nano Second: %lfns\n", nanoSec);
}
void load()
{
    int n = 1, fd;
    char c[10];
    fd = open("file", O_RDONLY);
    while (n--)
    {
        read(fd, c, 0);
    }
    printf("%s", c);
}

void s_call2()
{
    printf("My pid is %d.\n", (int)getpid());
```

```
54 }
55
56 void s_call3()
57 {
58     int rc = fork();
59     if (rc < 0)
60     {
61         exit(-1);
62     }
63     if (rc)
64     {
65         printf("\t--ls result--\n\n");
66         wait(NULL);
67         printf("\n\n\t--ls result--\n");
68     }
69     else
70     {
71         execl("/bin/ls", "ls", NULL);
72     }
73
74     return 0;
75 }
76
77 int main()
78 {
79     uint64_t start;
80     // Measure 1
81     start = m_start(1);
82     load();
83     m_end(start);
84     // Measure 2
85     start = m_start(2);
86     s_call2();
87     m_end(start);
88     // Measure 3
89     start = m_start(3);
90     s_call3();
91     m_end(start);
92 }
```

Its execution results are as follows:

```
1      ----------Measure Start1----------
2      Cycle : 1411425
3      Second: 0.006721s
4      Nano Second: 672107.142857ns
5      ----------Measure Start2----------
6      My pid is 6189.
7      Cycle : 167688
8      Second: 0.000799s
9      Nano Second: 79851.428571ns
10     ----------Measure Start3----------
11             --ls result--
12
13     hw2.c  makefile  output
14
15
16             --ls result--
17     Cycle : 12025222
```

```
18      Second: 0.057263s
19      Nano Second: 5726296.190476ns
```

Here we can see the result have 3 Measure. One is to call open and read and it cost 0.006721s in my computer. The second is to call getpid and it cost 0.000799s in my computer. The third is to call ls and it cost 0.057263s in my computer.