CSCI3150 – Tutorial 4

INTRODUCTION TO TIME

Calvin Kam <hckam@cse.cuhk.edu.hk>

Agenda

1. What is time in system?
2. Process Time – User Time + System Time
3. Alarm()

Time

In kernel, time is measured in different ways:

1. Real Time
   - Human understandable time
   - Used when interacting with users
2. Process Time

   Time in execution of a process in CPU.
   1. User Time – time spent on user code
   2. System Time – time spent on the kernel working on process's behalf.

Clock ticks

There are two clocks in the computer:

- Hardware clock and **software clock.**

The kernel keeps a clock itself via the software clock, which is maintained in kernel in terms of periodic timer.

The timer will notify the system in a specified period of time, called ***clock ticks.*** It is measured in *frequency (Hz).*

In Linux, we can get the clock ticks per second by:

```
long x = sysconf (_SC_CLK_TCK);
```

times() system call

How can we get the user time and system time of a process?

We can use: **`clock_t times(struct tms *buf);`**

times() retrieves the process time of the running process and its children in **clock ticks.**

There is a special data structure for this, as defined in <sys/times.h>

```
#include <sys/times.h>
struct tms {
clock_t tms_utime; /* user time consumed */
clock_t tms_stime; /* system time consumed */
clock_t tms_cutime; /* user time consumed by children */
clock_t tms_cstime; /* system time consumed by children */
};
```

times() system call

```
1  int main(int argc,char *argv[]) {
2      struct tms timebuf;
3      double ticks_per_sec = (double)sysconf(_SC_CLK_TCK);
4      int i;
5      printf("Going to loop 1000000000....\n");
6      for(i = 0;i < 1000000000;i++);
7
8      times(&timebuf);
9      printf("utime:%f\n",timebuf.tms_utime / ticks_per_sec);
10     printf("stime:%f\n",timebuf.tms_stime / ticks_per_sec);
11     printf("Process Time: %f\n",(timebuf.tms_utime +
   timebuf.tms_stime ) / ticks_per_sec);
12
13   return 0;
14 }
```

times() system call

User time →
System time →

```
$./1-GetProcessTime
Going to loop 1000000000....
utime:2.980000
stime:0.000000
Process Time: 2.980000
```

As the times() system call returns clock ticks, remember to convert it to seconds!!!!

Alarm()

If you want to be notified after a short interval, **alarm()** system call can help.

```
#include <unistd.h>

unsigned int alarm(unsigned int seconds);
```

If the alarm sounds, it will deliver **SIGALRM** to the process.

The default behavior is Termination. Therefore we will need to write a handler by ourselves ☺

Alarm()

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

void handler(int signum)
{
  printf("Dingggggggg.\n");
}
int main(int argc,char *argv[])
{
  signal(SIGALRM,handler);
  alarm(5);
  pause();
  return 0;
}
```

Custom Signal Handler

Alarm()

```
$ ./2-Alarm
Waiting for the signal
Dingggggggg.
```

After calling **alarm(),** the program is not blocked.

To observe the effect, another system call **pause()** is called.
It blocks the process until a signal comes.

```
#include <unistd.h>

int pause(void);
```

End

Stay tuned for the assignment 1 series :)