ECE437/CS481

M02E: PROCESSES & THREADS CLOCK & TIMING

Xiang Sun

The University of New Mexico

Process Timing

- ❑ Different concepts of time

  - ✓ User time of a process: the amount of CPU time spent in user-mode (outside the kernel) for the process. This is only actual CPU time used in executing the process. Other time spend (e.g., the time of the process being blocked) do not count.

  - ✓ System time of a process: the amount of CPU time spent in kernel-mode for the process. Other time spend (e.g., the time of the process being blocked) do not count.

  - ✓ Real time/Wall clock time of a process: overall time from start to terminate of the process.

  - ✓ Real time ? User time+ System time

Process Timing

❑ Timing --- When? Get the wall-clock time

- Use system call: gettimeofday--retrieve the current Coordinated Universal Time (UTC)

```
#include < sys/time.h>
int gettimeofday(struct timeval *tp, struct timezone *tzp);
```

- ✓ It sets the system's notion of the current time
- ✓ A timeval structure includes the following members:

```
long tv_sec; /* seconds */
long tv_usec; /*microseconds */
```

- ✓ A timezone structure includes the following members:

```
int tz_minuteswest; /* minutes west of Greenwich */
int tz_dsttime; /* type of dst correction */
```

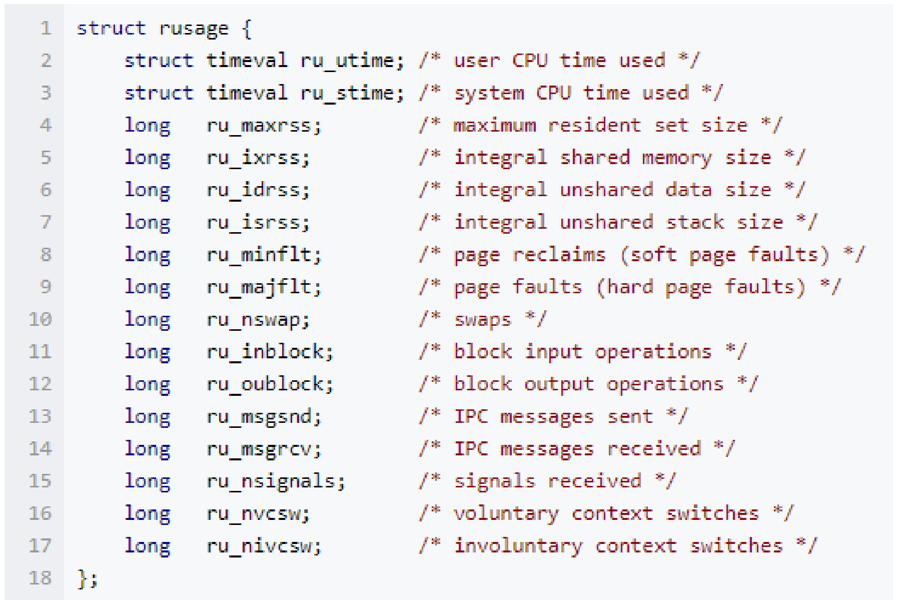
Process Timing

❑ Timing --- How long? Measuring processing time

- ➢ We might be interested in the CPU's execution time rather than the calendar time.
- ➢ Use system call: *getrusage*

```
#include <sys/time.h>
#include <sys/resource.h>
int getrusage(int who, struct rusage *usage);
```

- ✓ who: RUSAGE_SELF, RUSAGE_CHILDREN, RUSAGE_THREAD
- ✓ The rusage structure contains:



```
struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long   ru_maxrss;        /* maximum resident set size */
    long   ru_ixrss;         /* integral shared memory size */
    long   ru_idrss;         /* integral unshared data size */
    long   ru_isrss;         /* integral unshared stack size */
    long   ru_minflt;        /* page reclaims (soft page faults) */
    long   ru_majflt;        /* page faults (hard page faults) */
    long   ru_nswap;         /* swaps */
    long   ru_inblock;       /* block input operations */
    long   ru_oublock;       /* block output operations */
    long   ru_msgsnd;        /* IPC messages sent */
    long   ru_msgrcv;        /* IPC messages received */
    long   ru_nsignals;      /* signals received */
    long   ru_nvcsw;         /* voluntary context switches */
    long   ru_nivcsw;        /* involuntary context switches */
};
```

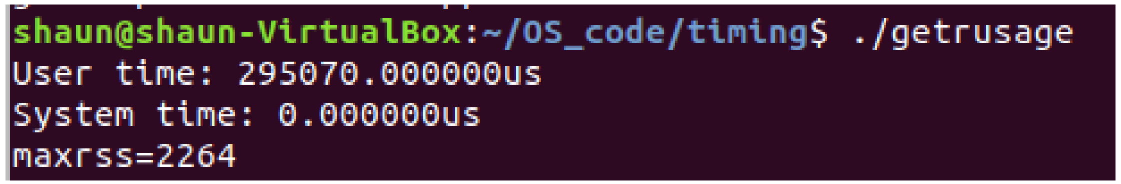
The accuracy of getrusage is us ($1*10^{-6}$ s)

Process Timing

❑ getrusage() example: obtain the user time and system time of the "for" loop

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <math.h>

int main(int argc, char** argv){
        double dum, usertime, systime;
        struct rusage r;
        struct timeval u_start, s_start, u_end, s_end;
        getrusage(RUSAGE_SELF, &r);
        u_start=r.ru_utime;
        s_start=r.ru_stime;
        for(int i = 0; i < 100000000; i++){dum=i*exp(0.5)+i;}
        getrusage(RUSAGE_SELF, &r);
        u_end=r.ru_utime;
        s_end=r.ru_stime;
        usertime=(u_end.tv_sec*1000000+u_end.tv_usec)-(u_start.tv_sec*1000000+u_start.tv_usec);//us
        systime=(s_end.tv_sec*1000000+s_end.tv_usec)-(s_start.tv_sec*1000000+s_start.tv_usec); //us
        printf("User time: %fus\n", usertime);
        printf("System time: %fus\n", systime);
        printf("maxrss=%ld\n", r.ru_maxrss);
        return 0;
}
```



```
shaun@shaun-VirtualBox:~/OS_code/timing$ ./getrusage
User time: 295070.000000us
System time: 0.000000us
maxrss=2264
```

Process Timing

- ❑ Timing --- How long? Measuring processing time
  - ➢ We might be interested in the CPU's execution time rather than the calendar time.
  - ➢ Use system call: times

```
#include < sys/times.h>
clock_t times(struct tms *buffer);
```

- ✓ times() stores the current process times in the struct tms that buffer points to.
- ✓ It returns the number of clock ticks that have elapsed since an arbitrary point in the past.
- ✓ clock_t is arithmetic types capable of representing times.
- ✓ Convert clock_t to second in times(): (clock_t) value/sysconf(_SC_CLK_TCK).
- ✓ The tms structure, defined in < sys/times.h>, contains the following members:

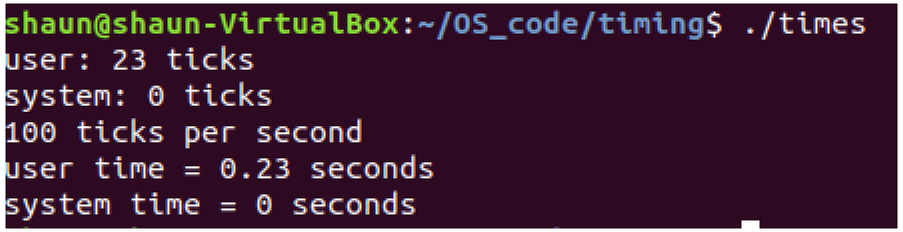
```
clock_t tms_utime; /* user time */
clock_t tms_stime; /* system time */
clock_t tms_cutime; /* children's user time */
clock_t tms_cstime; /* children's system time */
```

Process Timing

- times() example

```c
#include <stdio.h>
#include <sys/times.h>
#include <unistd.h>
#include <limits.h>

main () {
    struct tms buf;
    clock_t sinceboot;
    // chew up some CPU time
    int i,j;
    for (i=0,j=0; i<100000000; i++) { j+=i*i; }
    // measure elapsed time
    sinceboot = times(&buf);
    printf("user: %ld ticks\n", buf.tms_utime);
    printf("system: %ld ticks\n", buf.tms_stime);
    int ticks_per_second = sysconf(_SC_CLK_TCK);
    printf("%d ticks per second\n",ticks_per_second);
    double usec = (double)buf.tms_utime / (double) ticks_per_second;
    double ssec = (double)buf.tms_stime / (double) ticks_per_second;
    printf("user time = %g seconds\n", usec);
    printf("system time = %g seconds\n", ssec);
}
```



```
shaun@shaun-VirtualBox:~/OS_code/timing$ ./times
user: 23 ticks
system: 0 ticks
100 ticks per second
user time = 0.23 seconds
system time = 0 seconds
```

Process Timing

- Timing --- Put process to sleep
  - Use C-library call: sleep

```
#include <unistd.h>
unsigned int sleep(unsigned int seconds);
```

- ✓ The current process is suspended from execution for the number of seconds specified by the argument.
- ✓ The actual suspension time may be less than that requested--any caught signal will terminate the sleep().
- ✓ The suspension time may be longer than requested because of the scheduling of other activity in the system.
- ✓ The value returned by sleep() will be the amount of ``un-slept" time (the requested time minus the time actually slept)

Process Timing

- Timing --- Put process to sleep
  - Use C-library call: usleep

```
#include <unistd.h>
int sleep(useconds_t useconds);
```

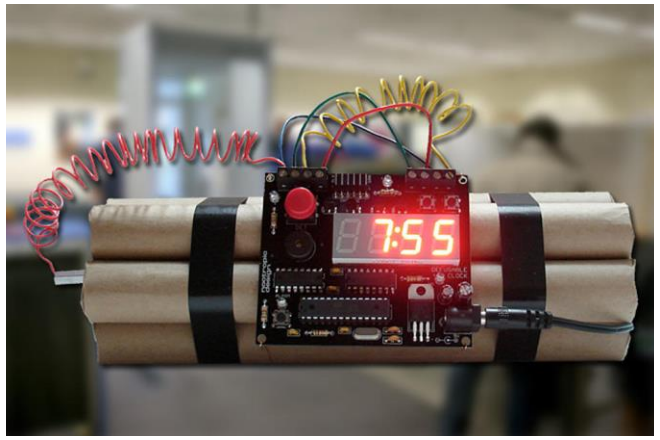
- ✓ The usleep() function suspends the current process from execution for the number of microseconds specified by the useconds argument.

Process Timing

❑ Timing --- Set alarm timer

➢ Use C-library call: alarm

```
#include <unistd.h>
unsigned int alarm(unsigned int sec);
```



- ✓ The alarm() function arranges for a SIGALRM signal to be delivered to the calling process in sec seconds.
- ✓ If sec=0, any pending alarm is canceled.
- ✓ alarm() returns the number of seconds remaining until any previously scheduled alarm was due to be delivered, or zero if there was no previously scheduled alarm.
- ✓ If the SIGALRM is not caught, blocked, or ignored by the calling process, the calling process will be terminated.