Agenda

- Four methods for measuring time in Linux
  - Three in GNU C
- Exercise
  - “Warm-up” question
  - “Instructor” question
  - For “quicker” results, work in pairs

Measuring time

time

- Basic: `time EXECUTABLE`
  - Example:
    ```
    $ time ls
    a.out
    real 0m0.002s
    user 0m0.000s
    sys 0m0.000s
    ```

time(1)

- Advanced: `/usr/bin/time [options] EXECUTABLE`
  - Example:
    ```
    $ /usr/bin/time --verbose ls
    a.out
    Command being timed: "ls"
    ...
    ```
  - Use of format:
    ```
    /usr/bin/time -f "%C,%U,%S,%E" ls
    ```

GNU C Library - time(2)

- #include <time.h>
- `time_t time(time_t *t);`
- Returns number of seconds since “Epoch”, i.e., Jan/01/1970 00:00:00 UTC
- `time_t` is usually signed integer
- Example:
  ```
  t1 = time(NULL);
  ```

GNU C Library - gettimeofday(2)

- #include <sys/time.h>
- Example: `gettimeofday(&t1, NULL);`
  - `t1` receives number of secs and usecs since Epoch
  - `t1` is a structure “timeval”:
    ```
    struct timeval {
    time_t tv_sec;       /* seconds */
    suseconds_t tv_usec; /* microsecs */
    }
    ```
  - `gettimeofday` returns success or error (0 or -1)
RDTSC

- Stands for “Real Time Stamp Counter”
  - x86 instruction since Pentium
- Counts clock cycles
  - EAX receives low-order 32 bits
  - EDX receives high-order 32 bits
  - Rate: /proc/cpuinfo
- Out-of-order execution
  - Use CPUID to force serialization

Example

```c
unsigned long long int RDTSC(void)
{
    unsigned int low, hi;
    asm volatile ("cpuid
	" "rdtsc
	": "=a"(low), "=d"(hi)::);
    return ((unsigned long long)hi << 32)|low;
}
```

Exercise

- Question: Estimate your CPU Frequency, i.e., how many clock cycles per second.
- Hints:
  - man sleep(3)
  - Write a C function that returns that information after a number of measurements (you might need that soon).

Exercise

- Question: How long does it take to read and write 1KB of data?
- Instructions:
  - Compare results using command time, measurement in microseconds and measurement using clock cycles.
  - Perform 100 reads and writes
  - Obtain mean and standard deviation for each method
  - Suggestion: Read from /dev/urandom and write to /dev/null

It's practice time!