Running Real-Time Tasks in Linux (x86 and ARM)

Sisu Xi
CSE 467S, Spring, 04/14/2014
Real-Time Tasks

- Task: a sequence of jobs
  - Period
  - Worst case execution time
  - Deadline (usually equals period)

- Scheduler
  - Static priority: RM
  - Dynamic priority: EDF
  - Preemptive vs. non-preemptive

1. Periodically triggered tasks?
2. Worst case execution time?
3. Scheduler setup?
4. Deadline miss?

A real time task with period of 5, execution time of 2, and deadline of 4
1. Periodically Triggered Task

- Video decoding, sensor processing, etc.
- [http://www.cse.wustl.edu/~xis/programs/period_task.c](http://www.cse.wustl.edu/~xis/programs/period_task.c)

```c
struct sigaction sa;

... 
sa.sa_sigaction = work;
sigaction(SIGRTMIN, &sa, NULL);

...
struct sigevent timer_event;
timer_event.sigev_signo = SIGRTMIN;

...
timer_create(CLOCK_REALTIME, &timer_event, &timer);
timer_settime(timer, TIMER_ABSTIME, &timespec, NULL);
```

- Many other approaches, see pointers
2. Workload for Tasks – Setup

- Minimize interference/uncertainties
  - Turn off unnecessary services
    - Bluetooth, network, graphic, etc
  - Fix CPU frequency
    - CPU frequency scaling
      [Link](https://wiki.archlinux.org/index.php/CPU_Frequency_Scaling)
    - CPU frequency governors
      [Link](http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt)
    - Disable in BIOS (SpeedStep for Intel, PowerNow for AMD)
    - Set to Performance/Custom/PowerSave
      - `ls /sys/devices/system/cpu/cpu0/cpufreq/
      - `echo performance > /sys/devices/system/cpu/cpu0/cpufreq/
governors`
2. Workload for Tasks – Programs

- **Simulated workload**
  - MiBench benchmarks [http://www.eecs.umich.edu/mibench/](http://www.eecs.umich.edu/mibench/)

- **CPU intensive workload**
  - Measure workload for 1 ms, then scale to any milliseconds workload
  - Note: need to tune for each individual machine
3. Scheduler Setup – Basic

Partitioned run queue, holds pending jobs
http://oreilly.com/catalog/linuxkernel/chapter/ch10.html

Two classes, would always schedule RT class first

- RT class: static priority, 1 (lowest) to 99 (highest)
  - SCHED_FIFO, SCHED_RR
  - Can be used to implement static priority (like rate monotonic)
  - SCHED_DEADLINE since Linux 3.14
  - Reserve 5% for other classes
    - /proc/sys/kernel/sched_rt_period_us 1000000
    - /proc/sys/kernel/sched_rt_runtime_us 950000

- Non-RT class: SCHED_OTHER with Complete Fair Scheduler
3. Scheduler Setup – Preemptive

- Scheduler is triggered every HZ quantum

- `cat /boot/config-* | grep CONFIG_HZ`
  - For most desktops, value is 1000. ticked every 1ms
  - For arm processors, value is usually 100. ticked every 10 ms

- `CONFIG_NO_HZ = y`

- `CONFIG_HIGH_RES_TIMERS = y`
  
  [http://elinux.org/High_Resolution_Timers](http://elinux.org/High_Resolution_Timers)

- Can recompile kernel to change these values
3. Scheduler Setup – Priorities

- chrt command (can also check task priorities)  
  - sudo chrt –f –p 99 4800  # pid 4800 with priority 99 and fifo

- sched_scheduler

```c
#include <sched.h>

int main() {
    ...
    struct sched_param sched;
    sched.sched_priority = 99;
    if (sched_setscheduler(getpid(), SCHED_FIFO, &sched) < 0) {
        exit(EXIT_FAILURE);
    }
    ...
}
```
3. Scheduler Setup – Affinities

- taskset command (can also check task affinities) [link]
  - sudo taskset -c 2,3 4800  # pid 4800 runs on cores 2-3

- sched_setaffinity [link]

```c
#include <sched.h>

int main() {
    ...
    unsigned long mask = 1;
    if (sched_setaffinity(getpid(), sizeof(mask), &mask) < 0) {
        exit(EXIT_FAILURE);
    }
    ...
}
```
4. Measure Deadline Miss

- gettimeofday()
  - return struct timeval, includes tv_sec and tv_usec
  - not ok for measuring overhead on standard kernel configuration

- rdtsc
  - read CPU cycles directly (need to fix CPU frequency)
  - cat /proc/cpuinfo to get CPU frequency
  - on a 3.33GHZ CPU, ticks 3,330,000,000 times per second
    - 3 tick equals 1ns
    - if you use rdtsc to record time, pay attention to this value
    - cat /proc/cpuinfo # get CPU frequency
    - cat /sys/devices/system/clocksource/clocksource0 # get current clock
Demo

- /proc/cpuinfo
- /proc/sys/kernel/sched_rt_*
- /sys/devices/system/clocksource/clocksource0/
- /sys/devices/system/cpu/cpufreq
- /sys/devices/system/cpu/cache/

- /boot/config*

- /dev/shm

- Compile demo task (g++ period_task.c -lrt)
Raspberry Pi

- ARM based Linux box
- Model B ($35)
  - 700 MHz, single core
  - 512 MB RAM
Raspberry Pi – CPU

- `cat /proc/cpuinfo`
  - ARMv6 compatible processor rev 7 (v61)

- `ls /sys/devices/system/cpu/cpu0/cpufreq`
  - cur, max, min freq
  - scaling_governor

- `uname --a`
  - Linux raspberrypi 3.6.11+ #371 PREEMPT Thu Feb 7 16:31:35 GMT 2013 armv6l GNU/Linux
  - Linux Preempt-RT patch: [https://rt.wiki.kernel.org](https://rt.wiki.kernel.org)
    - Minimize scheduling latency
Raspberry Pi – Clock

- `/sys/devices/system/clocksource/clocksource0/available_clocksource`
  - stc, software simulated clock, ticked every 1 microseconds

- `cat /proc/config.gz | gzip –d | grep HZ`
  - CONFIG_NO_HZ = y  # when idle, tickles to save energy
  - CONFIG_HZ = 100  # tick every 10 ms
Raspberry Pi – Summary

- Single core processor: no need to set cpu_masks
- Fixed frequency: no need to change frequency governor
- No rdtsc
  - Need to record time using other functions calls like gettimeofday, clock_gettime(), etc
  - Minimum time resolution provided is 1 microseconds via stc
  - Default scheduling tick is 10 ms (compared to 1 ms on Desktop)
- Comes with Preempt-RT patch by default
  - Less scheduling latency, more responsive to I/O tasks
Demo

- `/sys/devices/system/clocksource/clocksource0/available_clocksource`
- `/proc/config.gz | gzip –d | grep HZ`
- `/sys/devices/system/cpu/cpu0/cpufreq`
- `/proc/cpuinfo`
- `uname –a`

- Compile demo task
  - No rdtsc, need to replace with gettimeofday()
Note: Be careful with serial cable!

Overbaked my Raspberry Pi

Serial cable is supposed to be connected into Ground, GPIO 14 (TXD) and GPIO 15 (RXD).

Board stopped working when I connected the GPIO 14 cable into the GPIO 4 (CPCLK0) slot. Only the red power indicator comes on and nothing boots.
Pointers

- Periodically running a task
  - [http://www.cse.wustl.edu/~xis/programs/period_task.c](http://www.cse.wustl.edu/~xis/programs/period_task.c)
  - [http://www.embedded-linux.co.uk/tutorial/periodic_threads](http://www.embedded-linux.co.uk/tutorial/periodic_threads)

- Video players
  - [https://wiki.litmus-rt.org/litmus/Publications](https://wiki.litmus-rt.org/litmus/Publications)

- Get time in Linux
  - gettimeofday: [http://linux.die.net/man/2/gettimeofday](http://linux.die.net/man/2/gettimeofday)

- Fix CPU frequencies
  - [http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt](http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt)
  - [https://wiki.archlinux.org/index.php/CPU_Frequency_Scaling](https://wiki.archlinux.org/index.php/CPU_Frequency_Scaling)
Pointers

- **Linux schedulers**

- **Set priority**
  - sched_scheduler: [http://linux.die.net/man/2/sched_setscheduler](http://linux.die.net/man/2/sched_setscheduler)

- **Set CPU affinity on multi-core:**
  - taskset: [http://linux.die.net/man/1/taskset](http://linux.die.net/man/1/taskset)

- **Linux real-time patches:**
  - RTAI: [https://www.rtai.org/](https://www.rtai.org/)
  - SCHED_DEADLINE: [http://gitorious.org/sched_deadline](http://gitorious.org/sched_deadline)
Pointers

Raspberry Pi

- clock: [http://blog.remibergsma.com/2013/05/12/how-accurately-can-the-raspberry-pi-keep-time/](http://blog.remibergsma.com/2013/05/12/how-accurately-can-the-raspberry-pi-keep-time/)
- source code: [https://github.com/raspberrypi/linux](https://github.com/raspberrypi/linux)
- recompile kernel: [http://elinux.org/RPi_KernelCompilation](http://elinux.org/RPi_KernelCompilation)