Real-Time Thread

RTOS Task

- NOT necessarily responsible for everything
- A big system can be partitioned into multiple functions running simultaneously
- One person can focus on only one function
- RTOS threads help this

RTOS Support

Interrupt Management
rtl_request_irq
rtl_free_irq
rtl_hard_enable_irq
rtl_hard_disable_irq

Time Management
clock_gethrtime
clock_gettime
clock_settime
gethrtime
nanosleep

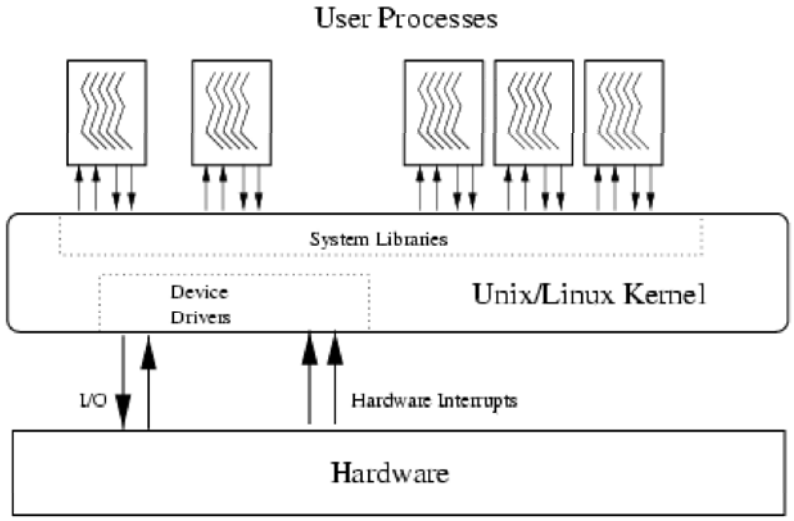
Task Management
pthread_create
pthread_setschedparam// pri. sched
pthread_make_periodic_np
pthread_wait_np
pthread_delete_np
pthread_cancel
pthread_join

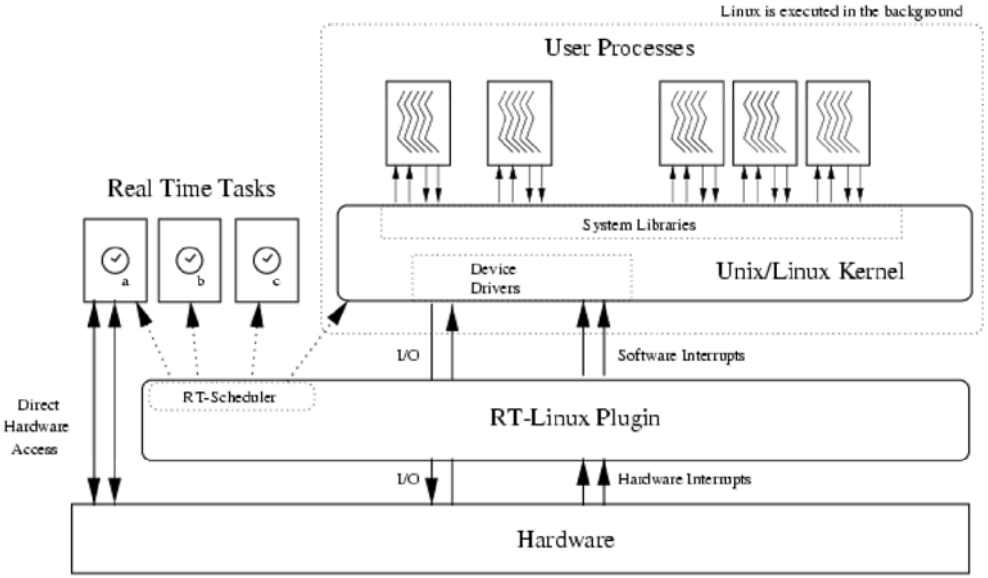
Task Communication
FIFO
Shared Memory
Signal

Mutual Exclusion
Lock
Semaphore

Device drivers
rt_com
rtsock

RtLinux Architecture





Periodic task with thread
First method (1)

- Create thread
  - In “int init_module()” ----- called by “insmod”
    - #include <pthread.h>
    - int pthread_create(pthread_t * thread, pthread_attr_t * attr, void * (*start_routine)(void *), void * arg);
  - How to transfer arguments (e.g., priority, period, etc.)?
- Run thread ------
- Delete thread
  - In “cleanup_module()” ----- called by “rmmod”
    - int pthread_cancel(pthread_t thread_id);
    - int pthread_join(pthread_t thread_id, void **thread_return);

Periodic task with thread First method (2)

- Set thread scheduling parameters
  - p.sched_priority = interger number;
    - larger number = higher priority
    - int sched_get_priority_min (int policy); // try SCHED_FIFO policy
    - int sched_get_priority_max (int policy);
  - int pthread_setschedparam(pthread_t thread_id, int policy, const struct sched_param *param);
    - pthread_t pthread_self(void);

Periodic task with thread
First method (3)

- Timing
  - make it periodic with “nanosleep”
    - #include <time.h>
    - int nanosleep(const struct timespec *req, struct timespec *rem);
    - struct timespec { time_t tv_sec; long tv_nsec; }
  - accurate time measure with “clock_gethrtime”
    - #include <rtl_time.h>
    - hrtime_t clock_gethrtime(clockid_t clock); // try CLOCK_REALTIME

Periodic task with thread First method (3)

- Example code (three periodic tasks)
  - One highest task periodically perform fibonacci(1000000)
  - Two low priority task periodically perform some nested loop
- How periodic?

```
pthread_t thread_id, thread_id1, thread_id2;

int init_module(void) {
        int arg[2];

        pthread_create (&thread_id, NULL, heavyThread, NULL);
        arg[0] = 0; arg[1] = 500000000; // 0.5 sec
        pthread_create (&thread_id1, NULL, thread, (void *)arg);
        arg[0] = 1; arg[1] = 800000000; // 0.8 sec
        pthread_create (&thread_id2, NULL, thread, (void *)arg);
        return 0;
}

void cleanup_module(void) {
        pthread_cancel(thread_id);
        pthread_join(thread_id, NULL);
        pthread_cancel(thread_id1);
        pthread_join(thread_id1, NULL);
        pthread_cancel(thread_id2);
        pthread_join(thread_id2, NULL);
}
```

```
void * heavyThread(void *arg)
{
        struct sched_param p;
        struct timespec sleep_time, rem;
        hrtime_t releaseTime, finishTime;

        sleep_time.tv_sec =0;
        sleep_time.tv_nsec = 100000000; // 0.1 sec

        p.sched_priority = sched_get_priority_max(SCHED_FIFO);
        pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);

        while (1) {
                nanosleep(&sleep_time, &rem);
                releaseTime =
                     clock_gethrtime(CLOCK_REALTIME);
                fibonacci(1000000);
                finishTime =
                    clock_gethrtime(CLOCK_REALTIME);
        }
        return 0;
}

void * thread(void *arg)
{
        ……
        ……
}
```

Periodic task with thread Second method (1)

- Make it periodic with
  - #include <rtl_sched.h>
  - int pthread_make_periodic_np(pthread_t * thread, hrtime_t start_time, hrtime_t period);
  - int pthread_wait_np(void);

Periodic task with thread Second method (2)

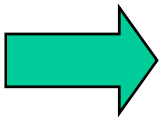
- Example code (three periodic tasks)
  - One highest task periodically perform fibonacci(1000000)
  - Two low priority task periodically perform some nested loop
- How periodic?

```
void * heavyThread(void *arg)
{
        struct sched_param p;
        struct timespec sleep_time, rem;
        hrtime_t releaseTime, finishTime;

        sleep_time.tv_sec =0;
        sleep_time.tv_nsec = 100000000; // 0.1 sec

        p.sched_priority = sched_get_priority_max(SCHED_FIFO);
        pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);

        while (1) {
                nanosleep(&sleep_time, &rem);
                releaseTime =
                     clock_gethrtime(CLOCK_REALTIME);
                fibonacci(1000000);
                finishTime =
                    clock_gethrtime(CLOCK_REALTIME);
        }
        return 0;
}
```



```
void * heavyThread(void *arg)
{
        struct sched_param p;
        hrtime_t releaseTime, finishTime;

        p.sched_priority = sched_get_priority_max(SCHED_FIFO);
        pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);

        pthread_make_periodic_np (pthread_self(), gethrtime(),
        100000000);

        while (1) {
                pthread_wait_np();
                releaseTime =
                    clock_gethrtime(CLOCK_REALTIME);
                fibonacci(1000000);
                finishTime =
                    clock_gethrtime(CLOCK_REALTIME);
        }
        return 0;
}
```

Project

- Implement a system with the following four periodic tasks
  - One second priority task periodically performs fibonacci(1000000) with period 0.1sec.
  - Two lowest priority tasks run stepping motor with speeds of 1 turn/sec and 2 turns/sec, respectively.
  - One highest priority tasks periodically checks the rotation count of the two motor at every 10 sec period (Hint! use global variable).
- Problem 1: measure and report worst case execution time of each task. Also, report the average execution time for large enough iterations.
- Problem 2: Write a program that checks the schedulability using “time-demand analysis” and “utilization bound check”. Answer if the system is schedulable.
- Problem 3: Implement the periodic tasks with “Sleep” method and “pthread_make_periodic_np” method. Each task should print out the time differences between expected release times and actual release times and between expected deadlines and actual finish times. Discuss the results.
- Problem 4: How to use “Sleep” method to make the periodic task more periodic? Implement your idea and report the results.

Project

- Compile and run prj2_schedule
- Explain what each task is doing and the results.
- Explain the schedule done by rtlinux scheduler
- Change the priority assignment in reverse-RM order. Explain the resulting schedule.
- Build a separate module which has one middle priority task (period of 143 ms), that periodically write “a, b, c, ….” to the global memory “mem”. At each period, the task put five characters to “mem” and report average execution time and max execution time. Can this module communicate with existing modules using the existing global variable?