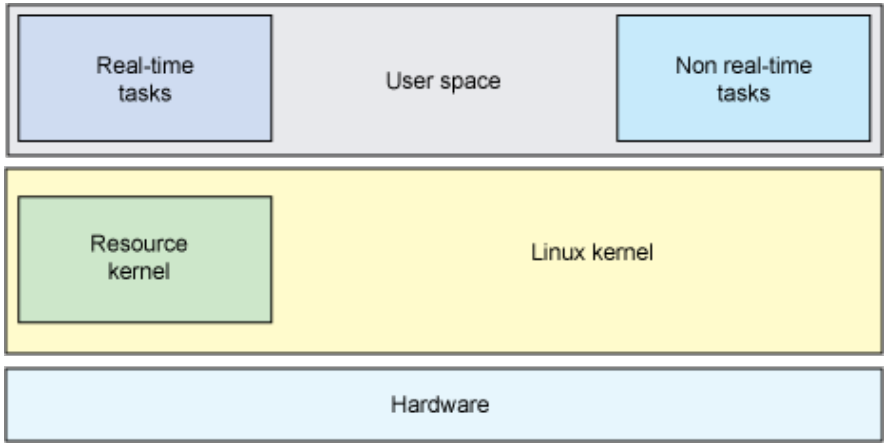
Lab 2 - Threads and Real Time Tasks

Week 2 – Real Time Tasks in User Space



ECE 4220/7220
Real Time Embedded Computing
Fall 2019
University of Missouri

What is a Linux scheduler?

- An important goal of a scheduler is to allocate CPU time slices efficiently while providing a responsive user experience.

- A CPU can be considered a resource to which a scheduler can temporarily allocate a task (in quantities called *slices* of time).

- The scheduler makes it possible to execute multiple programs at the same time, thus sharing the CPU with users of varying needs.

Why Use Real Time Tasks for Scheduling?

- Processes are not limited by the general purpose GNU/Linux scheduler
- Processes can be executed precisely as designed
- Meet hard real time constraints

Common Functions

int timerfd_create(**int** *clockid*, **int** *flags*)

- This function creates a new timer object, and returns a file descriptor that refers to that timer.
- The *clockid* argument specifies the clock that is used to mark the progress of the timer, and must be either **CLOCK_REALTIME** or **CLOCK_MONOTONIC**.
- The *flags* argument should be set to **0**.

int timerfd_settime (**int** *fd*, **int** *flags,* **const struct itimerspec** **new_value,* **struct itimerspec** **old_value*)

- This function arms (starts) or disarms (stops) the timer referred to by the file descriptor fd.
- The *fd* argument specifies the file descriptor from timerfd_create().
- The *flags* argument should be set to **0**.
- The *new_value* argument specifies the initial expiration and interval for the timer, a *struct itimerspec*.
- The *old_value* argument should be set to NULL.

Common Functions

int sched_setscheduler(pid_t *pid*, **int** *policy*, **const struct sched_param** **param*)

- This function sets both the scheduling policy and the associated parameters for the process whose ID is specified in *pid*. If *pid* equals zero, the scheduling policy and parameters of the calling process will be set.
- The *policy* argument specifies the scheduling policy that the task should follow, possibilities are **SCHED_OTHER**, **SCHED_BATCH**, **SCHED_IDLE**, **SCHED_FIFO** (Realtime), and **SCHED_RR** (Realtime).
- The *param* argument specifies a *struct sched_param* with the correct *sched_priority* value.

ssize_t read(int ***f****d*, **void** *buf, **ssiza_t** *count*)

- This function attempts to read up to count bytes from file descriptor *fd* into the buffer starting at *buf*.

Helpful Links

- Functions you need to use
  - https://linux.die.net/man/2/timerfd_create
  - https://linux.die.net/man/2/sched_setscheduler
  - https://linux.die.net/man/2/read