WAIT QUEUE

Harin Chandu
CDAC-HYD
rharin@cdac.in

OUTLINE

- What is kernel sleeping mechanism?
- Why ?
- What are wait queues?
- How to implement wait queues?
- How Process going to sleep
- How the process wakeup

The kernel sleeping mechanism

1. The sleeping is mechanism by which a process relaxes a processor.
2. The possibility of handling another process

The reason why a processor sleeps

1. It could be for sensing data availability or
2. Waiting for a resource to be free.

Wait queue

- There are several ways of handling sleeping and waking up the process in Linux, each suited to different needs.
- Wait queue is a mechanism provided in the kernel to implement the wait
- wait queue is the list of processes waiting for an event. In other words, A wait queue is used to wait for someone to wake up when a certain condition is true.
- **They must be used carefully to ensure there is no race condition**.

How to Implement Wait queues

There are 3 important steps, to implement Wait Queue.

1. Initializing Wait Queue
2. Queuing (Put the Task to sleep until the event comes)
3. Waking Up Queued Task

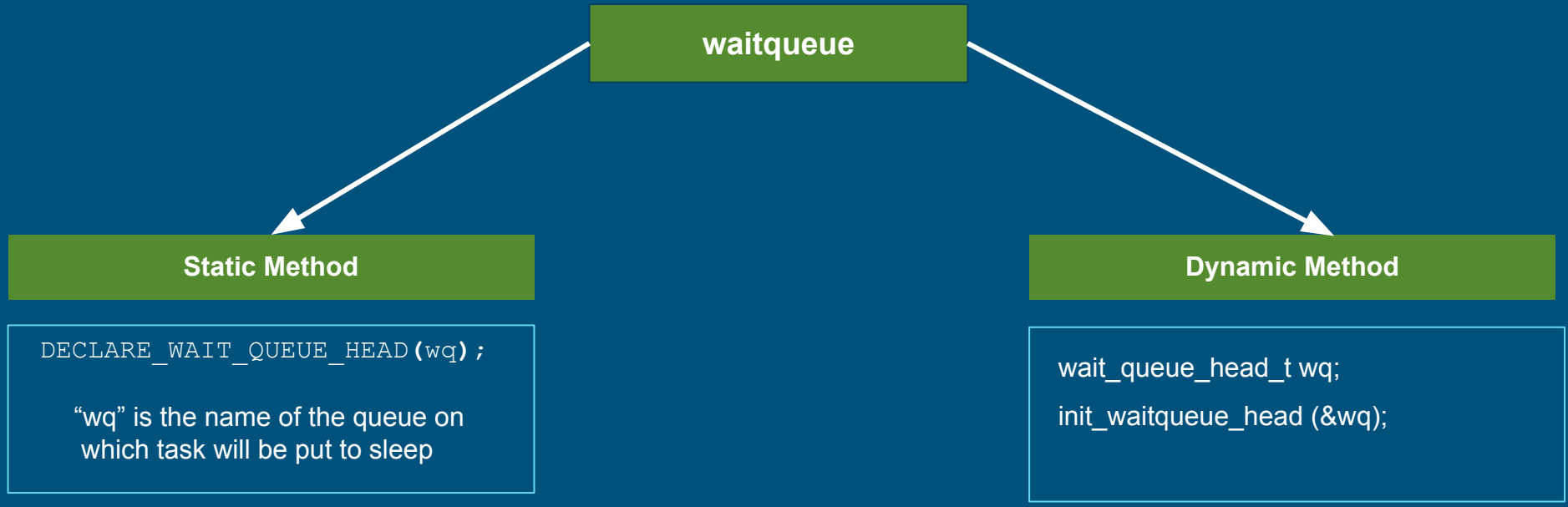
Initializing waitqueue

There are two ways to initialize the wait queue.

1. Static method
2. Dynamic method

Header file for Waitqueue (`include /linux/wait.h`)

Initializing wait queues



Process Going to Sleep

- **Wait_event**
  The process is will go sleep when a condition gets true(task uninterruptible)
  wait_event(wq, condition);

- **Wait_event_timeout**
  The process is will go sleep when a condition gets true or a timeout elapses (task uninterruptible)
  wait_event_timeout(wq, condition,timeout);

- **Wait_event_cmd**
  The process is will go sleep when a condition gets true (task uninterruptible)
  wait_event_cmd(wq, condition,cmd1,cmd2);
  - **Cmd1 :** the command will be executed before sleep
  - **Cmd2 :** the command will be executed after sleep

Process Going to Sleep cont…

- **Wait_event_interruptible**
  The process is to put to sleep when a condition gets true (task interruptible)
  wait_event_interruprible(wq, condition);
  The function will **return -ERESTARTSYS** if it was interrupted by a signal and 0 if **condition** is true.

- **Wait_event_interruptible_timeout**
  The process is to put to sleep when a condition gets true or a timeout elapses (task interruptible)
  wait_event_interruprible_timeout(wq, condition, timeout);
  The function will **return -ERESTARTSYS** if it was interrupted by the signal. 0 when the time elapsed.

- **Wait_event_killable**
  The process is to put to sleep when a condition gets true (task killable)
  wait_event_interruprible_killable(wq, condition);
  The function will **return -ERESTARTSYS** if it was interrupted by the signal.

Waking Up Queued Task

- **Wake_up**
  wakes up only one process from the wait queue which is in non-interruptible sleep.
  wake_up(&wq);

- **Wake_up_all**
  Wakes up all the processes on the wait queue
  wake_up_all(&wq);

- **Wake_up_interruptible**
  Wakes up only one process from the wait queue that is in interruptible sleep
  wake_up_interruprible(&wq);

THANK YOU