Project Ideas

- Requirements
  - Implementation
  - Make a presentation
  - Make a web-page about w/ source code, technical details
- Circuit Board Layout
- Block mode Device Drivers
- M-BOX Device Driver/Air Trombone
- M-BOX Ehancements: .WAV (Sound Bite) and .MIDI Modes
- (Real Time) Java on the Cerf Board
- 8051 – I2C network “stack” and application (Chat?)
- Other Ideas are welcome

The Reentrant Sonar Driver

Partitioning – A Case Study

- User Process – no control over when it executes. Can't access system resourced directly.
- Device Driver – Provides file I/O interface to system resources (I/O, RAM, etc.)
- Interrupt System – Provides real-time response
- Robot Collision Avoidance System
  - Sensors
    - Electronic Compass and Sonar Range Finder
  - Low Priority (Highest level) of Processing
    - Knowing where it is and deciding how to get where its going
  - Time Critical Processing
    - Sonar Travel Time
  - **High Priority Processing**
    - **Avoid Crashing into things**
  - Partition into Linux user process, device driver, interrupt system

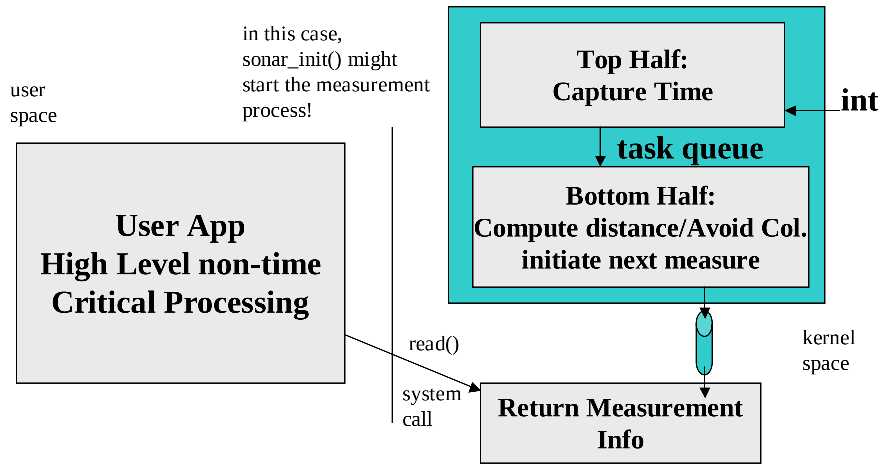
User Process

```
main () {
    while (1) {
     read (sonar, distance, n);
     read (compass, direction, n);
     here = where(here, distance, direction, speed, angle);
     move(here, there, &speed, &angle);
    }
}
```

- What do the device drivers do?
  - Should it be different than what we discussed so far?
  - Compass – Read proper memory mapped location to get current compass output and return result.
- Where do we do collision detection and avoidance??
  - The essence of a "soft" real time constraint.
  - We don't want to disable other time critical operations (ISR)
  - We can't let it run at the user level

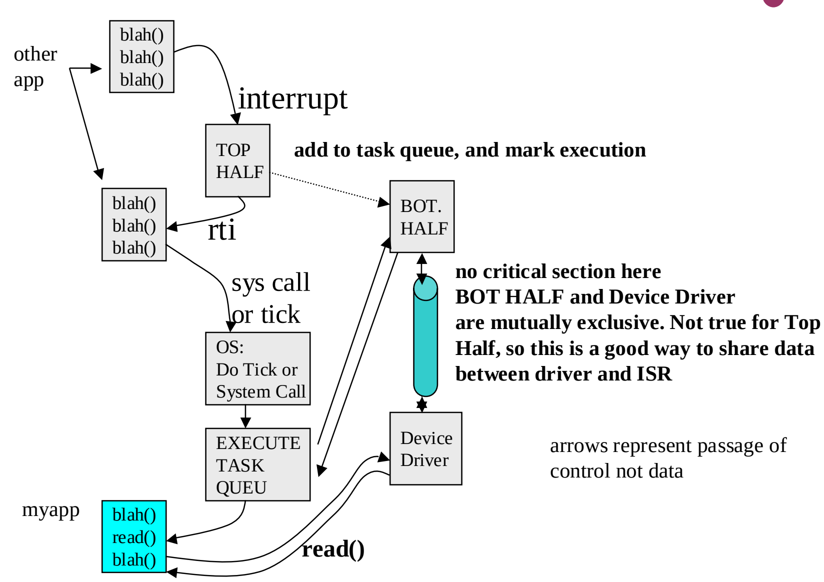
Answer: In the ISR...sort of

- Linux Nomenclature for Interrupt handling
  - Top Half – the actual ISR. Time critical stuff
  - Bottom Half – the time dependent, less critical stuff. Signaled by the ISR



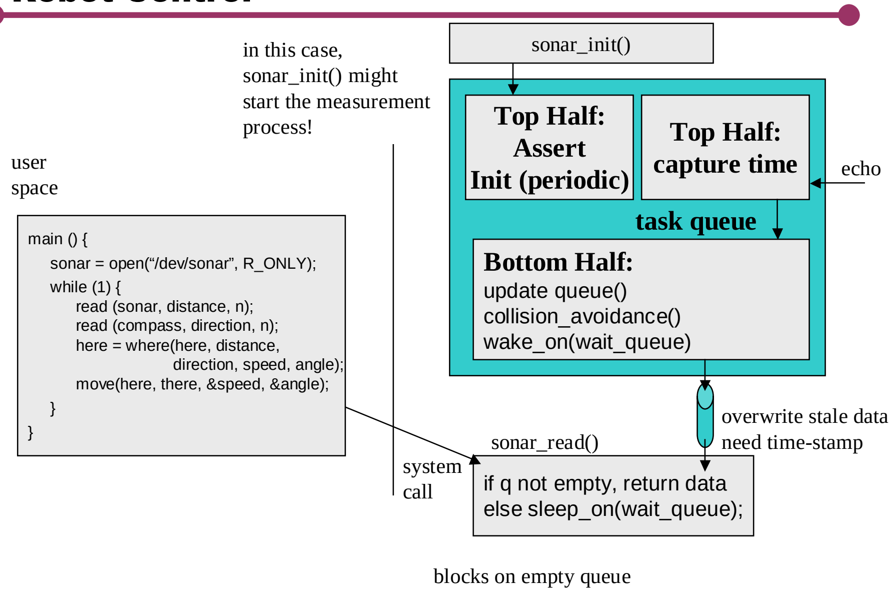
CSE 466 – Fall 2000 - Introduction - 5

Schedule of task queues...bottom halves in Linux

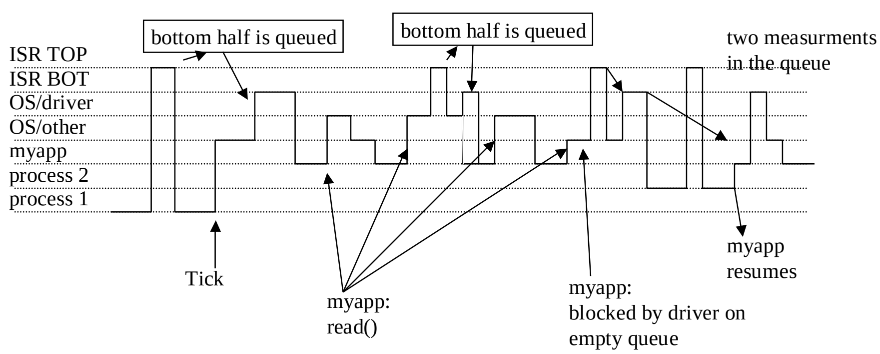


CSE 466 – Fall 2000 - Introduction - 6

Robot Control



As a Task Diagram



ISR BOT and Driver are mutually exclusive, so no problem with shared data structures.
Bottom half has manay opportunities to run
Worst case is the tick interval of the operating system. Can be changed in the Linux Kernel

Linux Interprocess Communication

Pipes and Fork()

```
void main() {
      int pfds[2];
      pipe(pfds);
      if (fork()) producer();
      else consumer();
}
void producer() {      // serial?
      int q = pfds[0];
      while (1) {generate_data(buf); write(q, buf, n);}
}
void consumer() {
      int q = pfds[1];
      while (1) {read(q, buf, n); process_data(buf);}
}
```

Single Reader, Single Writer

Kernel ensures mutual exclusion (Read/Write are system calls)

FIFO's, which are named pipes

- Process 1

```
void main() {
      mknod("/tmp/myfifo", S_IFIFO, );                // create a FIFO file node
      f = open("/tmp/myfifo", O_WRONLY);
      while (1) {generate_data(buf); write(q, buf, n);}
}
```

- Process 2

```
void main() {
      f = open("/tmp/myfifo", O_RDONLY);
      while (1) {read(q, buf, n); process_data(buf);}
}
```

- Works for "unrelated" processes
- Multiple writers, Multiple readers
  - Kernel ensures mutual exclusion
  - Kernel does not control interleaving of writers, readers

Impement a FIFO w/ a Device Driver

CSE 466 – Fall 2000 - Introduction - 11

Shared Memory

- Can we do this with a device driver?
- There are dedicated systems calls to support shared memory which are more efficient than device drivers

CSE 466 – Fall 2000 - Introduction - 12

Message Queues

- Like FIFO’s but for data structures rather than bytes (Structured I/O)
  - System calls
    - Create a message queue
    - Put messages on the message queue(q, message_pointer, message_type)
    - Get messages from the queue(q, message_pointer, message_type);
- It’s a really ugly in C
  - The data structure is serialized, but you can’t tell what the type is.
  - Sender and Receiver have to agree on integer designations for data types)
  - Java (and maybe C++) is sooo much better at this kind of thing

Implement w/ Shared Memory

- FIFO
- Shared Memory

Sockets -- Networking

- s = socket(int domain, int type, int protocol) -- protocol is usually associated with domain but not always
  - domain: internet, UNIX, apple-talk…etc. How to interpret that address
- bind(int s, sockaddr *addr, n)
  - s is the socket identifier
  - sockaddr is the address (june.cs.washington.edu)
  - n is the length of the address
- Now it is like a pipe, you can do read/write, send/recv, listen/accept.
- Several types
  - stream
  - datagram
  - sequential
  - raw
- The protocol stack
  - mapping from application level abstractions (open, read, socket) to HW and back