Föreläsning 6 IS1300 Inbyggda system

- Real time problem
  - Hard and soft deadline
  - Sequence versus concurrence programming
  - Processes
- Real Time Operating System
  - MicroC/OSII (used in lab exercise)

Diverse

- Embeddedpriset.nu
- Schemaändring Onsdag 6 februari em

Deadlines

- Hard deadline
  - System failure if deadline fails
- Soft deadline
  - Degraded performance if deadline fails

Give examples!

Sequence vs concurrent programming

- Sequential Programming
  - Sequence of actions that produce a result
  - Is called a process, task or thread
- Concurrent Programming
  - Two or more processes that work together
  - Need synchronisation and communication

Constructs needed for concurrent programming

- Notion of processes to express concurrent execution
- Process synchronization
- Communication between processes

  Via shared memory and/or by message passing

Concurrent programming

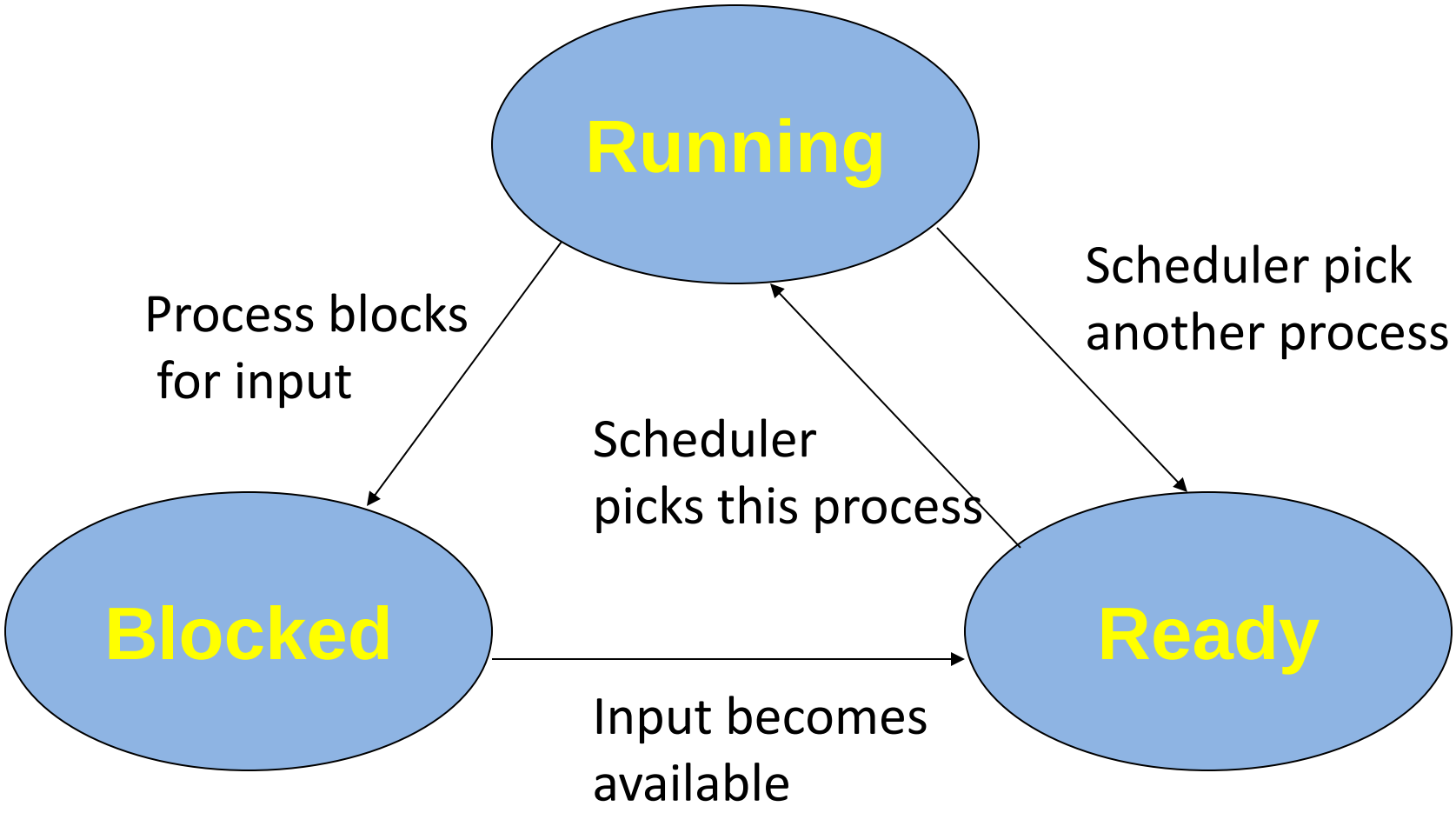
Usually takes one of three forms.

- Single processor
  - processes multiplex their executions on a single processor
- Multiprocessor
  - processes multiplex their executions on a multiprocessor system where there is access to shared memory
- Multicomputer (Distributed System)
  - processes multiplex their executions on several processors which do not share memory

Process

- A process is an executing sequential program
- Each process has its own virtual CPU

Process states



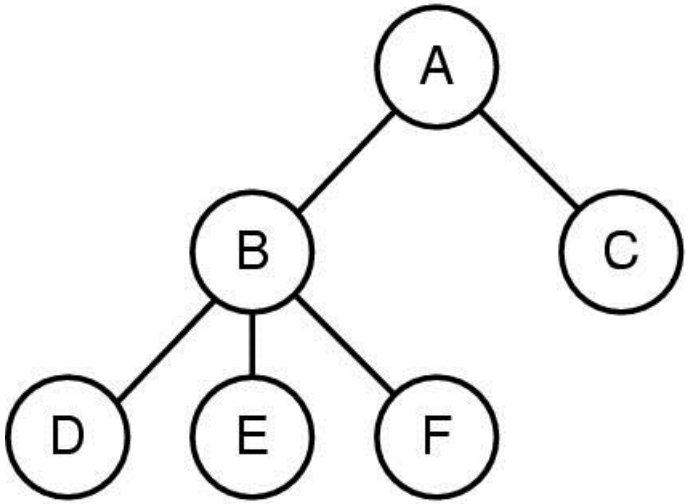
Processes

- Processes can be
  - Independent
    - Not synchronized or communicating
  - Cooperating
    - Synchronized and communicating
  - Competing
    - Peripheral devices, memory, and processor power
    - Must communicate to fairly share resources.

Hierarchy of processes

- A process can create a child process
- The child process can create new processes
- Parent / Child
  - Parent responsible for the creation of Child process
- Guardian / Dependent
  - The guardian process can not terminate until all dependent processes have terminated.

Hierarchy of processes



- A process tree
  - A creates two child processes, B and C
  - B creates three child processes, D, E, and F

Concurrent Programming in OS

Two main categories

- Pre-emptive multitasking
  The OS controls which process is executing.
- Co-operative multitasking
  The current process decides if it shall stop executing.

MicroC/OS-II

```
int main(int argc, char *argv[])
{
    INT8U  err;


    OSInit();  /* Initialize "uC/OS-II, The Real-Time Kernel" */

    OSTaskCreate(AppStartTask,
                 (void *) 0,
                 (OS_STK*) &AppStartTaskStk[TASK_STK_SIZE-1],
                 TASK_START_PRIO);

    OSStart(); /* Start multitasking (i.e. give control to uC/OS-
II) */
}
```

MicroC/OS-II

```
void  AppStartTask (void *p_arg)
{
  p_arg = p_arg;

  while (TRUE) /* Task body, always written as an infinite
loop. */
  {
    OS_Printf("Delay 1 second and print\n");
    OSTimeDlyHMSM(0, 0, 1, 0);
    /* OSTimeDly(1000) */
  }
}
```