Process

Discussion session 3

1/30/2016
What is the process?

• A program in execution
• An instance of a program running on a computer
• The entity can be assigned to and executed on a processor
• A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system instructions.
Process Model

- Multiprogramming of four programs
- Conceptual model of four independent, sequential processes
- Only one program active at any instant.
Process states

• Running : executing
• Blocked: waiting for I/O
• Ready: waiting to be scheduled.

• 1: process blocks for input
• 2: scheduler picks another process
• 3: scheduler picks this process
• 4: input becomes available.
Main OS process-related goals

• Interleave the execution of existing processes to maximize processor utilization
• Provide reasonable response times
• Allocate resources to processes
• Support inter-process communication (and synchronization) and user creation of processes
How are these goals achieved?

• Schedule and dispatch processes for execution by the processor
• Implement a safe and fair policy for resource allocation to processes
• Respond to requests by user programs
• Construct and maintain tables for each process managed by the operating system
When is a new process created?

- System Initialization (Daemons)
- Execution of a process creation system call by a running process
- A user request to create a process
- Initiation of a batch job
When does a process terminate?

• Normal exit (voluntary)
• Error exit (voluntary)
• Fatal error (involuntary)
• Killed by another process (involuntary)
Process snapshot

Virtual memory

Stack

Free space

Dynamic data

Static data

Code

PSW

Stack pointer

Program Counter
Process elements

- While the program is executing, this process can be uniquely characterized by a number of elements, including:
Process Table

- Collection of process descriptors for each task/process

- **Each entry contains:**
  - process identification()
    - Process ID, Parent’s ID, Children ID and Owner’s ID.
  - pointers to the Process Control Block (PCB) and process image for each process

**PCB ➔ Collection of attributes for managing the process**

- Process image
  - Collection of program, data (modifiable part), stack, and data stored in PCB

- A process may be linked to other processes which are related (e.g. children, parent)
Process Control Block (PCB)

• Identifiers
  • PID, PPID and User’s ID

• Process State Information
  • User-Visible Registers
    • Typically, 16 to 32 registers that can be referenced by user programs directly
  • Control and Status registers
    • Program counter: contains the address of next instruction
    • Condition codes: result of the most recent arithmetic or logical operation
    • Status information: interrupt enabled/disabled flags, execution mode, etc.

• Stack pointers:
  • each process has one or more LIFO system stacks which is used to store parameters and calling addresses for procedure and system calls.
Process Control Block (cont.)

- Process Control Information

- Scheduling and state information (needed by the OS to perform its scheduling function):
  - Process state: (e.g., running, ready, waiting/blocked, halted).
  - Priority: scheduling the priority of the process.
  - Scheduling-related information: that the scheduling algorithm may need (e.g. the amount of time spent in waiting, the amount of time used during the last time it was running, etc.)
  - Event: Id of event the process is awaiting before it can run again

- Data structuring:
  - A process may be linked to other processes in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent-child (creator-created) relationship with another process. The PCB may contain pointers to other processes to support these structures.
Process control block (cont.)

• Interprocess communication info
  • Various flags, signals, and messages associated with communication between processes.

• Process privileges
  • Processes are granted privileges in terms of the memory that may be accessed, the types of instructions that may be executed, the system utilities and services that may be used, etc.

• Memory management
  • includes pointers to segment and/or page tables that describe the virtual memory assigned to this process.

• Resource ownership and utilization
  • Resources controlled by the process such as opened files. A history of utilization of the processor or other resources may also be included (may be needed by the scheduler)
Process Creation

• Assign a unique process identifier (PID)
• Allocate space for the process.
• Initialize process control block.
• Set up appropriate linkages.
  • Ex: add new process to linked list (i.e. queue) used for scheduling

When to Switch a Process

• Clock interrupt
• I/O interrupt
• Memory (page) fault
  • The address referenced is not in main memory so it must be brought into main memory
• Trap
  • When an error occurred.
  • May cause process to move to the Exit state.
Execution within user processes

Figure 3.16 Process Image: Operating System
Executes Within User Space
Mode Switching

If no interrupts are pending the processor:

- proceeds to the fetch stage and fetches the next instruction of the current program in the current process.

If an interrupt is pending the processor:

- sets the program counter to the starting address of an interrupt handler program.
- switches from user mode to kernel mode so that the interrupt processing code may include privileged instructions.
Change of process state

• The steps in a full process switch are:

If the currently running process is to be moved to another state (Ready, Blocked, etc.), then the OS must make substantial changes in its environment.
Change of process state

- Save context of processor including program counter and other registers
- Update the PCB of the process that stopped running
- Move PCB to an appropriate queue - *ready, blocked*, etc.
- Select another process for execution
- Update the PCB of the process selected (e.g. change state to *running*)
- Restore context of the selected process
Exercise 2
Modifying the Process Table
The Process Table

Minix has two process tables: proc & mproc

- **proc**: kernel data structure
- **mproc**: userland data structure

The two are kept in sync by the kernel

The mproc table is used to generate procfs
Tasks

1. Add a new pid_t field to struct mproc in servers/pm/mproc.h

2. Modify servers/pm/forkexit.c to set the new field to double the value of the pid. Look at do_fork() and do_srv_fork()

3. Modify servers/procfs/pid.c to output the value. Look at pid_psinfo(int i)

4. Recompile, install, and reboot
Submission

On Canvas, Submit a screenshot of the two results of

```
cat /proc/<id>/psinfo
```

e.g.

```
cat /proc/108/psinfo
cat /proc/142/psinfo
```

There should be a visible field equal to \( 2 \times <\text{id}> \)

Deadline:
- Monday – 2pts
- Tuesday – 1pt