CS 450 Module R2
PCB Operations

West Virginia University

Spring 2023

WestVirginiaUniversity.

Goals

Goals

- Establish Process Control Blocks (PCBs) for tracking and managing processes
- Emplace PCBs in queues to enable scheduling
- Implement commands for managing PCBs

Background

Process Control Blocks 1/2

- For multiprocessing, MPX needs to track processes with these attributes
- Name – A user readable string capable of storing at least 8 characters, unique to each process
    - Process names can have a limit, but that limit has to be at least 8 characters
    - This does **not** mean process names need to be at least 8 characters
- Class – Indicates whether a process us a user application or system process
- Priority – An integer between 0 (highest priority) and 9 (lowest priority)
- State – Two data points
    - Execution state – ready, running, or blocked
    - Dispatching state – suspended or not suspended

Process Control Blocks 2/2

- Stack – A process stack area of at least 1024 bytes
    - **Not** a separate data structure, just memory space for the CPU stack
    - The stack should initialized to all binary 0
- Stack Pointer – A pointer to the current location (in the Stack) used by CPU instructions
    - On x86, the CPU stack grows *down*, so stack pointer should initially point to the last few bytes (enough to hold a *void **) of the stack
- Related PCBs – For building queues, you'll likely want pointers for creating singly- or doubly-linked lists

Process Queues

- Used to track processes in similar states
- Ready processes should be sorted by priority (low-to-high), then FIFO (first in, first out) for PCBs of the same priority
- Blocked processes should be in simple FIFO order
- Implementation
    - Option 1: Ready and Blocked queues
    - Option 2: Ready, Blocked, Suspended-Ready, and Suspended-Blocked queues
    - Option 3: Everything in one big queue
    - Option 4: Surprise me

Kernel Functions

`struct pcb* pcb_allocate(void);`

- Uses `sys_alloc_mem()` to allocate memory for a new PCB, including the stack, and perform reasonable initialization
- Parameters:
    - None
- Returns:
    - A pointer to a newly allocated PCB on success
    - `NULL` on error during allocation or initialization

`int pcb_free(struct pcb*);`

- Uses `sys_free_mem()` to free all memory associated with a PCB, including the stack
- Parameters:
    - A pointer to the PCB to free
- Returns:
    - A code indicating success or error

`struct pcb* pcb_setup(const char *, int, int);`

- Allocates (via `allocate_pcb()`) a new PCB, initializes it with data provided, and sets the state to Ready, Not-suspended
- Parameters:
  - The process name
  - The process class (user application or system process)
  - The process priority (0-9)
- Returns:
  - A pointer to the initialized PCB on success
  - `NULL` on error allocating, initializing, or invalid parameter

`struct pcb* pcb_find(const char *);`

- Searches all process queues for a process with the provided name
- Parameters:
  - The name of the process to find
- Returns:
  - A pointer to the found PCB on success
  - `NULL` if the provided name was not found

`void pcb_insert(struct pcb*);`

- Inserts a PCB into the appropriate queue, based on state and (if Ready) priority
- Parameters:
  - A pointer to the PCB to enqueue
- Returns:
  - None

`int pcb_remove(struct pcb*);`

- Removes a PCB from its current queue, but does not free any associated memory or data structures
- Parameters:
    - A pointer to the PCB to unqueue
- Returns:
    - A success or error code

User Commands

Create PCB

- Calls `pcb_setup()` to create a PCB, then inserts it into the appropriate queue with `pcb_insert()`
- Parameters:
  - Process Name
  - Process Class
  - Process Priority
- Error Checking:
  - Name must be unique and valid
  - Class must be valid
  - Priority must be valid

Delete PCB

- Finds the requested process, removes it from its queue with `pcb_remove()`, and frees all associated memory with `pcb_free()`
- Parameters:
  - Process Name
- Error Checking:
  - Name must be valid
  - Must **not** be a System process

Block PCB

- Puts a process in the blocked state, and moves it the appropriate queue
- Parameters:
  - Process Name
- Error Checking:
  - Name must be valid

Unblock PCB

- Puts a process in the unblocked (ready) state, and moves it the appropriate queue
- Parameters:
    - Process Name
- Error Checking:
    - Name must be valid

Suspend PCB

- Puts a process in the suspended state, and moves it the appropriate queue
- Parameters:
    - Process Name
- Error Checking:
    - Name must be valid
    - Must **not** be a System process

Resume PCB

- Puts a process in the not suspended state, and moves it the appropriate queue
- Parameters:
    - Process Name
- Error Checking:
    - Name must be valid

Set PCB Priority

- Changes a process's priority, and moves it to the appropriate place in the appropriate queue
- Parameters:
    - Process Name
    - New Priority
- Error Checking:
    - Name must be valid
    - Priority must be valid (0-9)

Show PCB

- Displays a process's:
    - Name
    - Class
    - State
    - Suspended Status
    - Priority
- Parameters:
    - Process Name
- Error Checking:
    - Name must be valid

Show Ready

- For all processes in the Ready state, display the process's:
  - Name
  - Class
  - State
  - Suspended Status
  - Priority
- Parameters:
  - None
- Error Checking:
  - None

Show Blocked

- For all processes in the Blocked state, display the process's:
    - Name
    - Class
    - State
    - Suspended Status
    - Priority
- Parameters:
    - None
- Error Checking:
    - None

Show All

- For all processes (in any state), display the process's:
    - Name
    - Class
    - State
    - Suspended Status
    - Priority
- Parameters:
    - None
- Error Checking:
    - None

Final Notes

Final Notes

- Remember to update:
  - Commands: Version and Help
  - Documentation: User's and Programmer's Manuals; Contributions
- Provide clear error messages – Referencing the manual should not be necessary to interpret errors
- Vary your testing – Don't simply repeat the exact same steps every time
- Pro tip: If your PCBs have pointers for a linked list, you don't need a separate data structure for queues
- Use the debugger