Project Proposal: Implementing Process Statistics System Call and Heap Growth Tracking System Call

---

1. Introduction

Effective process management and equitable CPU scheduling are fundamental aspects of contemporary operating systems. xv6 is a lightweight Unix-like operating system that serves as an excellent environment for experimentation with kernel-level ideas. CPU scheduling is a fundamental aspect of contemporary operating systems. xv6 is a lightweight Unix-like operating system that serves as an excellent environment for experimentation with kernel-level ideas.

This project develops the following features that extend xv6:

1. **Process Statistics System Call** – exposes runtime information per process.

2. **Heap Growth Tracking System Call:** This feature will help to understand the behavior of the memory more easily

These two features both improve observability, fairness of CPU scheduling, and memory, and provide an opportunity for developing simple OS kernel features.

---

2. Project Objectives: We are doing the following task for our project work

- **System Call Development:** We are implementing getpinfo to return detailed process metrics.It gathers and returns information about processes currently running in the system.

- **PCB Enhancement:** Extend struct proc with fields for CPU ticks, waiting time, scheduling count, state, and ticket count. First goes to (proc.h), then struct proc, and will add new variables here.

- **Testing:** Validate functionality using user programs and workload comparisons.

---

3. Feature 1: Process Statistics System Call

Tasks: xv6 lacks tools to monitor per-process behavior. This call provides metrics for CPU usage, waiting time, and scheduling frequency. However, xv6 has no internal mechanisms to monitor the behavior of each process at runtime, and therefore it cannot directly display information such as how long a process uses the CPU, how long it waits in the ready queue, or how many times it has been scheduled to run.

Design: Here, we will conduct

- struct pstat fields:
  - pid: Process ID
  - ticks: CPU time
  - waittime: Waiting time
  - scheduled_count: Times scheduled
  - state: RUNNING/RUNNABLE/SLEEPING
- int getpinfo(struct pstat *p): Copies statistics of all processes to a user buffer.

Outcome: Provides real-time insight into process behavior and a foundation for scheduler testing. xv6 will finally be able to show **live**, **detailed information** about how **each process behaves** inside the system. This includes seeing how much **CPU time** a process consumes, how long it **waits** before running, how often it **gets scheduled**, its **current state**, and—if using lottery scheduling—how **many tickets** it has.

---

4. Feature 2: Heap Growth Tracking System Call

Motivation: Any operating system performs memory management functions, but this is limited in xv6. Programs can only grow or shrink their heap with the ‘sbrk()’ system call, and there is no built-in way to inspect all these changes over time. For this reason, it can be hard to debug when memory is involved, especially in apps that frequently allocate and free memory.

The **heapinfo()** system call facilitates this action by providing a simple way to check heap information. It provides the developer with information about the amount of memory the process is using, how the heap expands and contracts as the program executes, and whether there is any unusual or unexpected memory usage. This helps simplify debugging and enhances transparency into how the programs use dynamic memory.

Design:

The **heapinfo()** feature is designed by making a small, modular change to the XV6. The main idea is to keep a simple log of all heap adjustments through **sbrk()**.

- Extended **proc** structure**:**

A fixed-size array for heap events and counter for heap events will be added to store heap growth or shrink values for each process.

- Modify **sbrk():**

Every time a process calls sbrk(n), the value of n is recorded in the process's event log. Positive values indicate heap expansion, and negative values indicate heap shrinkage.

- Implement **heapinfo() System Call:**

This System Call will copy the recorded heap event log from a curnel space to a buffer provided by the user program using **copyout()**.

- **Minimal Kernel Changes:**

Only a few files will be updated, keeping the design simple, time-efficient, and easy to maintain.

Outcome:

The hapinfo system call will successfully add visibility into a process's heap changes and will provide a clear log of all the expansions and shrink operations which be made through sbrk(). If we successfully implement this feature, it will become a help for developers, which will help them to understand the memory behavior more easily. If this feature gets implemented successfully, this will improve the debugging, support learning, and enhance the overall usability of xv6 without affecting the existing functionalities.

---

5. Expected Deliverables

- Modified xv6 source code (system calls + scheduler).
- User-level test programs.
- Project report detailing design, implementation, and results.
- Live demo and presentation.

---

6. Conclusion

This project enhances the xv6 operating system with improved observability features and a proportional-based scheduling mechanism. By developing a system call for process statistics and a lottery-based scheduler, we enhance our understanding of operating system internals, collaborate with kernel development, and refine scheduling algorithms. Overall, this enables us to develop a more insightful and demonstrable system designed to facilitate learning and experimentation with modern OS design principles.