ILLINOIS TECH

College of Computing

# CS 450 Operating Systems Sample Questions

Yue Duan

# Understand fork() and exec()

- fork() creates a new process
  - the created child process is an exact copy of the parent except for the return value of the fork() call.
- exec() executes a program
  - it overwrites the data in the original program
  - the data is replaced with a running copy of the new program.

# CPU Scheduling Algorithms

- **FIFO** (first in first out)
  - a.k.a, FCFS (first come first serve)
- **SJF** (shortest job first)
  - schedule jobs in order of estimated execution time
- **RR** (round robin)
  - each job allowed to run for a quantum, then switch to the next job
- **SRJF** (shortest remaining job first)
  - preemptive version of SJF scheduling
- **Multi-level Queue Scheduling**
  - multiple ready queues based on job ‘type’

# Disk scheduling

- Consider that requests to read the following set of logical block numbers are enqueued to be served from a disk that has 100 logical blocks laid out sequentially from block 0 to block 99.

  {1, 22, 14, 72, 86, 32, 11, 66, 45, 80}

  Assume that the seek time in moving the disk arm head from logical block i to block j is proportional to |i – j|. Given that the arm head is currently positioned at block 75, what is the sequence in which the enqueued blocks will be read with SSTF?

# Disk scheduling

- {1, 22, 14, 72, 86, 32, 11, 66, 45, 80}, current head 75
  - 72
  - 66
  - 80
  - 86
  - 45
  - 32
  - 22
  - 14
  - 11
  - 1

# Inode calculation

- Consider a UNIX-style inode with 10 direct pointers, one single-indirect pointer, and one double-indirect pointer only. Assume that the block size is 8K bytes, and the size of a pointer is 4 bytes. How many blocks (including indirect blocks) are needed to address a file of size 10MB?
  - **10MB = 10240KB**
  - **first 80KB stored in 10 blocks, so remaining 10240 - 80 = 10160KB**
  - **10160KB/8KB = 1270, since a single-indirect block can store 8KB/4B = 20K, we only need 1 single-indirect block**
  - **so total # of blocks = 10 + 1 + 1270 = 1281**

# Disk Access for File Operations

- Explain the first eight steps in the timeline below for file read

| | data bitmap | inode bitmap | root inode | foo inode | bar inode | root data | foo data | bar data [0] | bar data [1] | bar data [2] |
|---|---|---|---|---|---|---|---|---|---|---|
| open(bar) | | | **read** | | | **read** | | | | |
| | | | | **read** | | | **read** | | | |
| | | | | | **read** | | | | | |
| read() | | | | | **read** | | | **read** | | |
| | | | | | **write** | | | | | |
| read() | | | | | **read** | | | | **read** | |
| | | | | | **write** | | | | | |
| read() | | | | | **read** | | | | | **read** |
| | | | | | **write** | | | | | |

# Disk Access for File Operations

- (1) read root inode to locate root data
- (2) read root data to lookup foo and find foo's inode
- (3) read foo inode to locate foo data
- (4) read foo data to find bar's inode
- (5) read bar inode to retrieve bar's metadata as part of file open operation
- (6) read bar inode to locate bar data
- (7) read bar data
- (8) write bar inode to update its last access timestamp

| | data bitmap | inode bitmap | root inode | foo inode | bar inode | root data | foo data | bar data [0] | bar data [1] | bar data [2] |
|---|---|---|---|---|---|---|---|---|---|---|
| open(bar) | | | **read** | **read** | **read** | **read** | **read** | | | |
| read() | | | | | **read** **write** | | | **read** | | |
| read() | | | | | **read** **write** | | | | **read** | |
| read() | | | | | **read** **write** | | | | | **read** |

# THANK YOU!