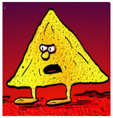


Operating Systems Project - NachOS

Team G - Defense

Azadeh Kakavand Christopher Ferreira Jaime Romero Vincent Xuereb

January 26, 2017

Table of contents

What we have done

Console

Kernel provides some functions for console input/output.
Calls are synchronous.
It is safe for concurrent threads.

Process and thread scheduling

Rely on the provided scheduling policy for NachOS threads.

Automatic termination

- Exit and Thread-Exit: Exit is called, we terminate the current thread.
- Halt: last process of system is exited, then the machine will be halted.

Threads

Different execution flows within the same address space

- Create
- Join
- Exit

Synchronization primitives

- Lock
- Condition Variable
- Read/Write Lock

Processes

Each process has its own :

- isolated address space
- set of live threads

Two operations

- **ForkExec** $\Rightarrow$ Create a process
- **WaitProcess** $\Rightarrow$ Wait for a process termination

File System

- Per-process working directory
- Hierarchy tree
- Directory navigation and operations
- Open file table
- Big files
- No support for growing files

Network

Reliable transfer of fixed-size message

- Connection-oriented
- Handshake
- Sequence number
- Acknowledgment

Network - What we are missing

- Variable size message
- File transfer

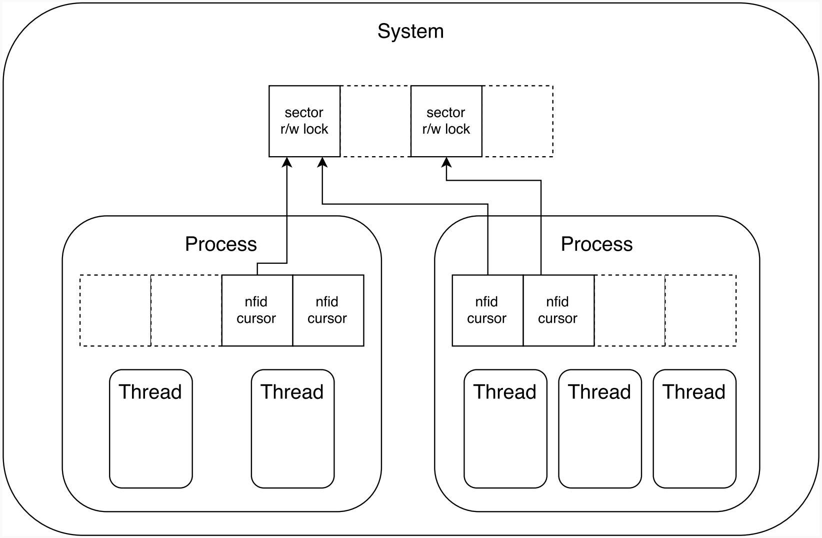
Unfulfilled wishes

File Synchronization

Granularity :

- Disk Sector
- FileRead/FileWrite Syscall
- User defined
- File open/close lifetime

File Synchronization



Miscellaneous

- Kernel-level thread safety
- User information on syscall failure
- User thread termination

Implementation insights

Unique ID

Two goals:

- Do not reuse ID
- Fast look-up

Unique ID

Two goals:

- Do not reuse ID
- Fast look-up

Solution:

- *index = ID % the maximum number of items*
- Use the next value whose cell is empty

Thread Stack Allocation

How to allocate user threads stack :

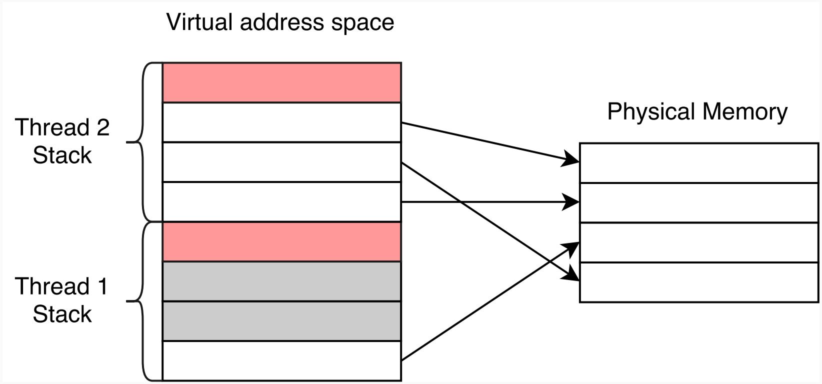
- Variable sizes depending on the number of threads
  $\Rightarrow$ Fragmentation issues

Thread Stack Allocation

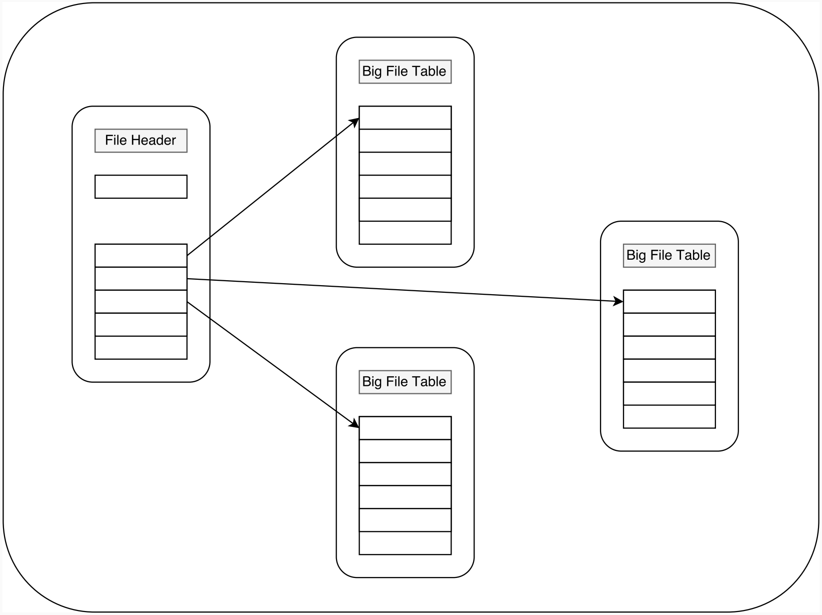
How to allocate user threads stack :

- Variable sizes depending on the number of threads
  $\Rightarrow$ Fragmentation issues
- Fixed size
  $\Rightarrow$ How to deal with different stack size requirements ?

Thread Stack Allocation



Big Files



Path Support

- Relative paths
- Absolute paths
- Paths for directory operations

Path Usage Examples

- CreateDirectory(**”dir1”**);
- CreateDirectory(**”/dir1/dir2”**);
- ChangeDirectory(**”../dir2”**);

Network

- A layer on top of the PostOffice
- Multiple connections per machine allowed
- Sleeping thread

Project Management

Organization

- GitLab issues
- Good communication
- No planning

No planning

- Tasks to stop/begin
- Time lost

Conclusion

?

- ??

Questions?