Operating System

# NachOS - Final Project Report

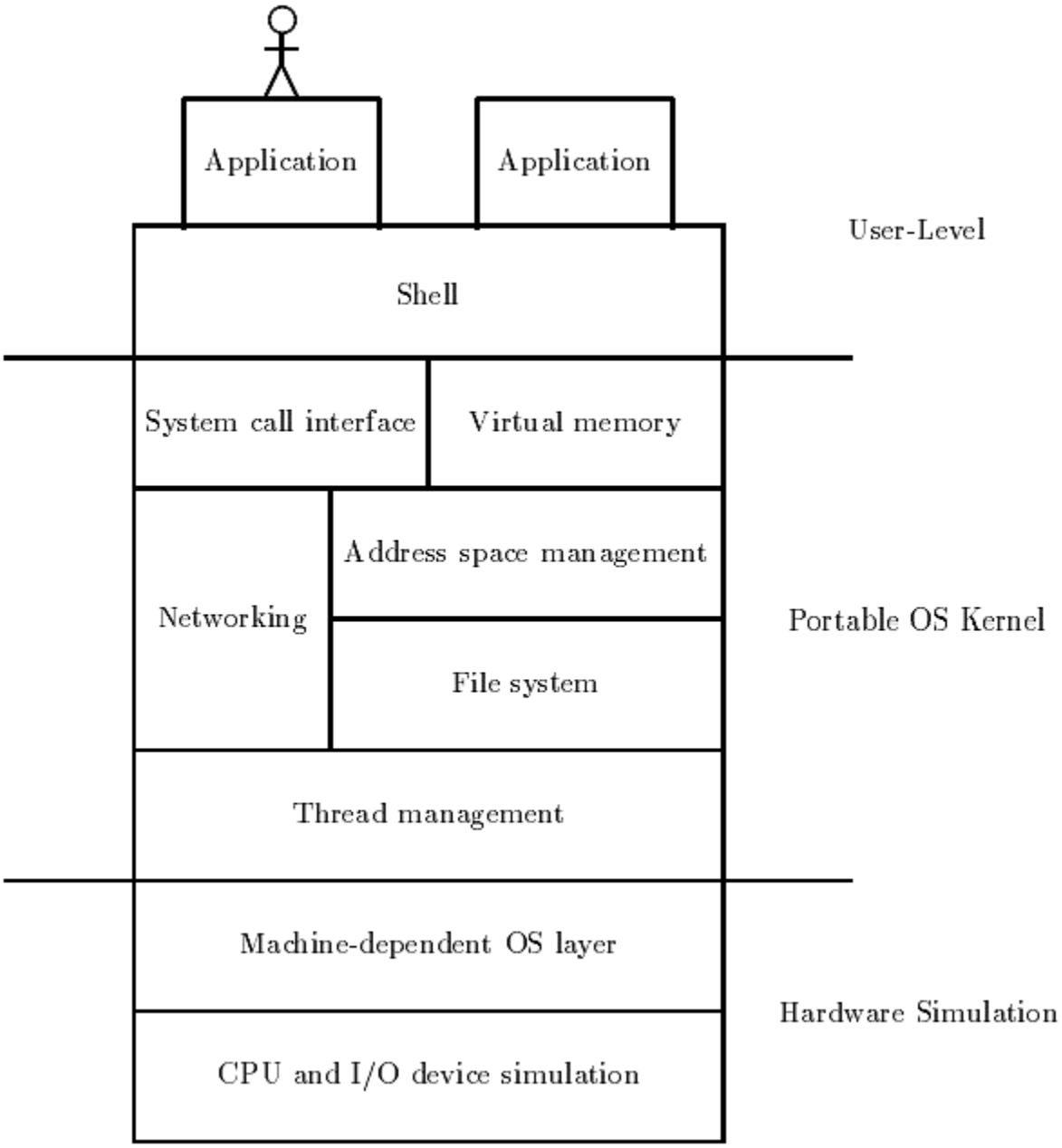
## Team Members

Neel Patel #201501075
Prasann Patel #201501083

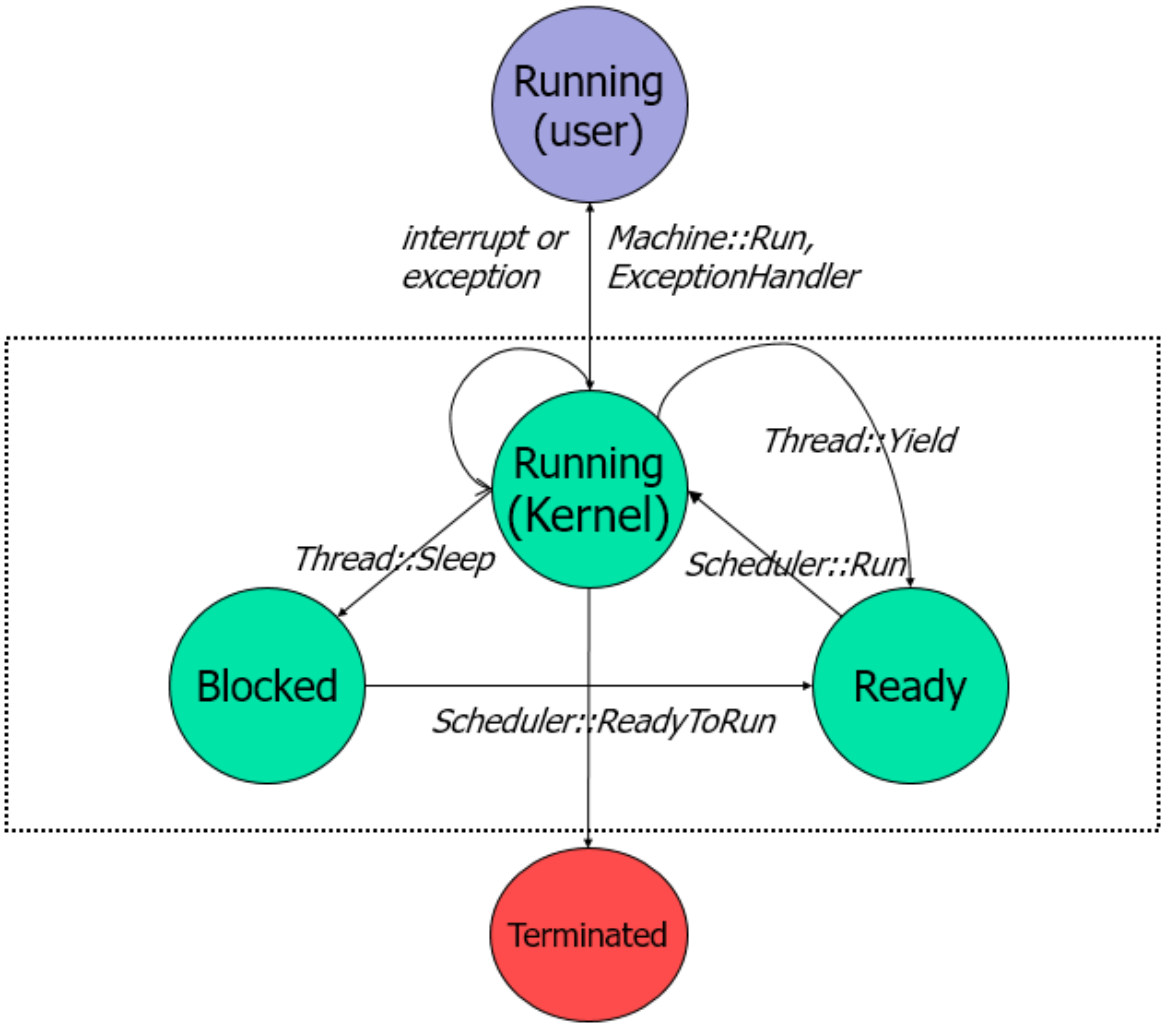
## Description

Not Another Completely Heuristic Operating System (NachOS) is a multitasking instructional system that runs on a UNIX process which illustrates and explores all areas of modern operating systems including threads and concurrency, multiprogramming, system calls, software-loaded TLB's, file systems, and distributed systems.

## Architecture



## Model/Diagram(s)




## Technical Specification

NachOS uses C++/JAVA and basic MIPS instructions to run as a user-process on top of host OS like Ubuntu-Linux. NachOS interacts with simulated H/W by calling functions that eventually calls underlying host OS library routines. Also, in NachOS' interrupts happen only as discrete points in real time as time is also simulated.

Scope of the project includes designing and implementing features like :

1. Thread management - Thread Control Block (TCB.contextSwitch)
2. Concurrency and Synchronization
3. Interrupt Handling
4. Multiprogramming - 50%
5. Filling services (i/o Console)

## Algorithm/Flow Chart

Pseudo-code for different functions in Kthread are as follows :

```
join() {
        Disable interrupts;
        if (join Queue not be initiated) {
        create a new thread queue (join Queue) with transfer priority flag opened
        join Queue acquires this thread as holder

        }
        If (CurrentThread != self) and (status is not Finished) {
                add current thread to join queue
                sleep current thread
        }
        Re-enable interrupts;
}
finish() {
        disable Interrupts.
        Destroy current thread
        Assign status = finished
        Sleep forever  }

sleep() {

        release condition lock
        disable interrupt
        add current thread to wait queue
        make current thread sleep
        restore interrupt
        acquire condition lock
}
```

***wake( )*** : wake up a single thread sleeping in this condition variable, if possible.

```
if wait queue is not empty
        disable interrupt
        remove the first element from wait queue
        put the first element into ready queue
        restore interrupt
```

***yield( )*** : Relinquish the CPU if any other thread is ready to run. If so put the current thread on the ready queue which will be re-scheuled later.

```
Store status of current thread
If wait queue is not empty
          Disable interrupts
          Send current thread to ready queue
          Invoke runNextThread( )
           restore interrupt
```

***run( )*** : Dispatch the CPU to this thread. Save the state of the current thread, switch to the new thread by calling TCB.contextSwitch() & load the state of the new thread. The new thread becomes the current thread.

**Note** : Ready never adds the idle thread to the running state [Eg. thread 1 (#ideal thread)]

```
Change state of previously running thread/ps.
disable interrupts
invoke Machine.yield( )
invoke tcb.contextSwitch( )
restore state
```

<u>KThread has status – New, Ready, Running, Blocked, Finished</u>

# Program and Test Data Set

Included in the source code file :

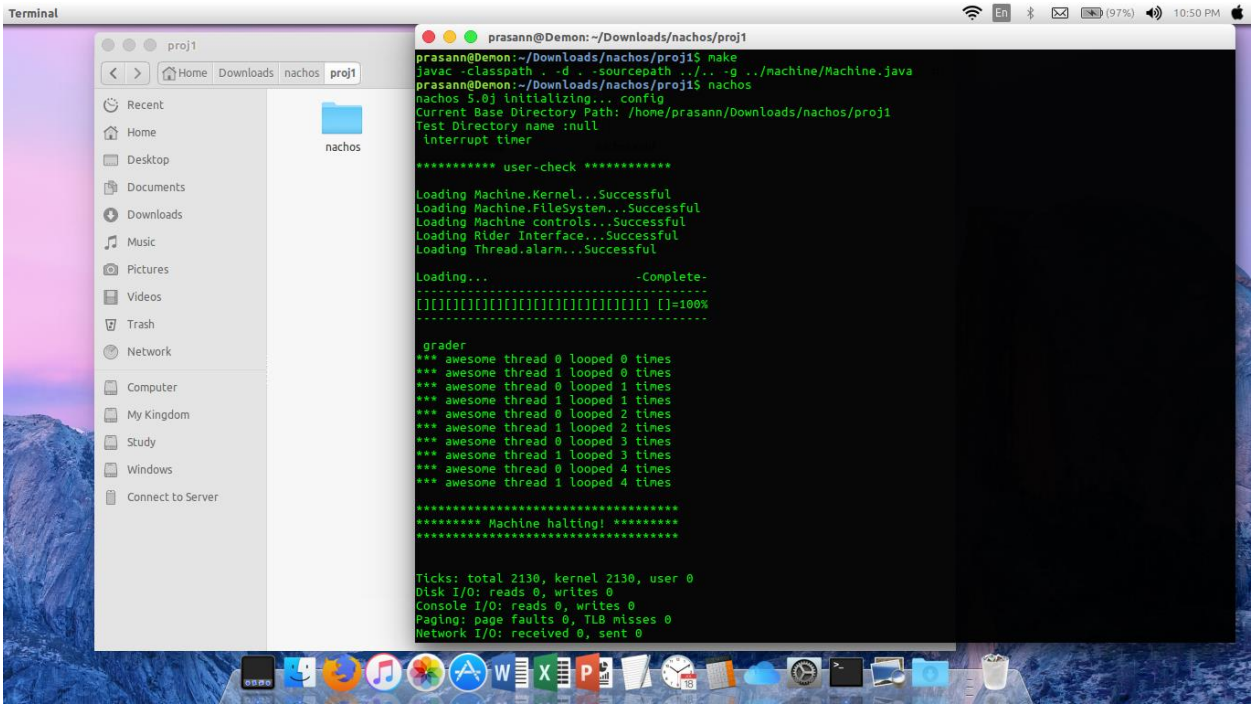
Test input :

1. cd nachos/proj#
2. make
3. nachos <switch\optional> <arguments>

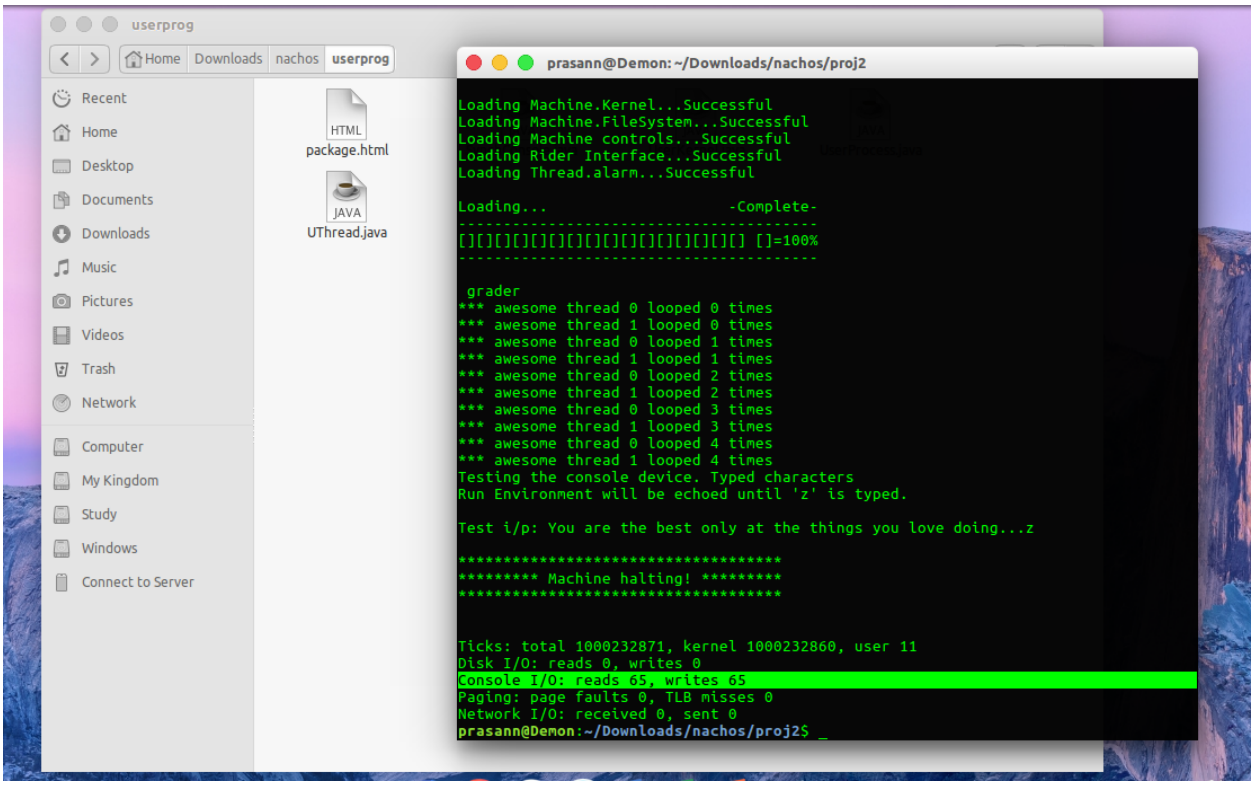
Eg. nachos -s 10 (random thread placement)
nachos -d t (to debug)
nachos (simple exe.)
nachos -np (About the developers and project)

# Results

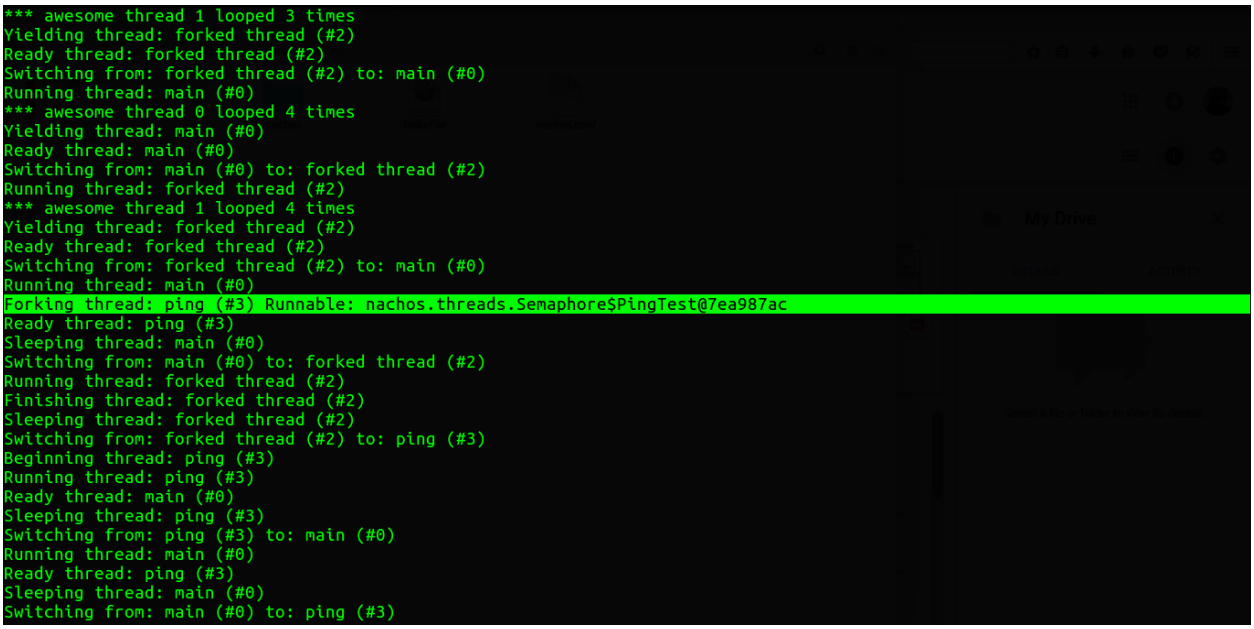
Proj1: Thread Management



Proj2: Multi-programing




Debug mode :



## Implementation/Source Code

**Attached in Code file.**

## References


1. *The Nachos Instructional Operating System* by Wayne A.Christopher, Steven J.Procter and Thomas E.Anderson -Computer Science Division -University of California - Berkeley

2. *University of California, San Diego : CSE 120 Nachos Project Guide - Fall 2017* http://cseweb.ucsd.edu/classes/fa17/cse120-ab/projects/index.html
3. CS162 Project Phase 1 : http://inst.eecs.berkeley.edu/%7Ecs162/fa10/Nachos/phase1.html