



# The Quest Kernel




Richard West and Gary Wong

Boston University




## Motivation




- Develop a relatively small kernel for research and educational purposes
  - Avoid complexities of existing open source systems
    - e.g., Linux is now millions of lines (including drivers)
    - Quest core has less than 7000 lines of C and assembly, with ~ 600-700 lines of assembly
      - Quest library < 1000 lines of C
  - Learn about design and implementation of systems on the "bare metal"
    - From bootstrapping to low-level device driver interaction
  - Use as a vehicle for researching new / novel mechanisms and policies on diverse hardware platforms

## Design (1 of 2)




- Quest currently works on x86 (32 bit)
- Features include:
  - Elementary shell (no significant CLI yet)
  - Support for IDE disks, keyboard, text-based terminal and VGA graphics
  - Limited implementation of EXT2 filesystem (taken from GRUB bootloader)
    - Mount, open, read filesystem features
  - Virtual memory (paging) capabilities
  - Fork / exec / exit process control semantics
  - O(1) preemptive priority queue scheduling
  - Minimalist "libc" system library
    - malloc / free ...

## Design (2 of 2)




- Quest leverages hardware-controlled task-switching on x86
  - Differs from software-based context switching on Linux
  - One "tss" per process / task in Quest unlike one per CPU in Linux
  - x86 "nested task" support for switchback to prior preempted task
    - e.g., terminal server and shell implemented as user-space tasks invoked via call gates
    - Possibility for user-space services ala  $\mu$ -kernels

## Main File Organization



- `boot.S, interrupt.S`
  - Assembly code for initializing IDT, syscall table, paging capabilities, memory layout (GDT for segments, initial kernel stack etc)
- `terminal_server.c`
  - User-space access capabilities to (text-mode) video RAM
- `shell.c, kernel.c`
- `sched.c, heap.c`
  - Policy-specific queue routines
- `init.c`
  - C final boot-stage code (called from `boot.S`)
  - ELF module code, PIC / PIT initialization etc
- `diskio.c`
  - Low-level IDE CHS/LBA block read/write code
- `interrupt_handler.c`
  - Syscall / interrupt handler bodies
- `fsys_ext2fs.c`
  - EXT2 filesystem support

## Example Applications



- Basic fork / exec / exit memory-leak tester
- Advent
  - 1970s text-based adventure game (ported from FORTRAN)
- Pacman
  - Mame-based implementation ☺
- NB: It's not a system if it can't play PACMAN!

## Sample Screenshots (1 of 2)

```

Bochs kernel version: 0.14
Copyright Boston University, 2005
***** 31649792 bytes free *****
~/boot/adventure
Initializing...
Couldn't find adventure.data, using adventure.text...

Table space used:
12482 of 12500 words of messages      879 of 885 travel options
326 of 330 vocabulary words          184 of 185 locations
67 of 180 objects                    34 of 35 action verbs
276 of 277 "random" messages         19 of 12 "Class" messages
10 of 20 hints                       4 of 5 turn thresholds

Welcome to Adventure!! Would you like instructions?
no
You are standing at the end of a road before a small brick building.
Around you is a forest. A small stream flows out of the building and
down a gully.
    
```

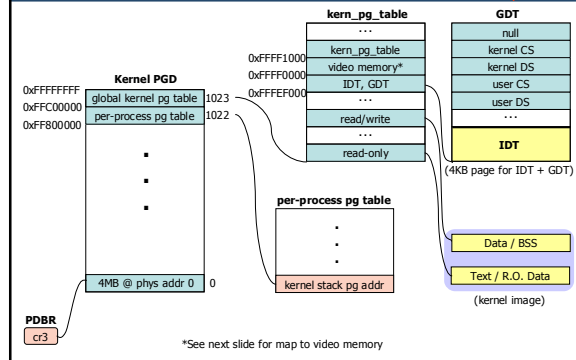
## Sample Screenshots (2 of 2)



## Quest Projects

- Port to ARM / XScale
- Add SMP support for x86 implementation
- Add ULS / Hijack executive features to Quest
- Develop new service management features
  - Scheduling / synchronization policies
  - Filesystem support
  - IPC mechanisms
- Look at interrupt management / accounting strategies
- Investigate shared cache policies (e.g., L2-aware scheduling)
- Add virtual machine interface (VMI) or user-space event delivery API

## Virtual Memory Layout (1 of 2)



## Virtual Memory Layout (2 of 2)

