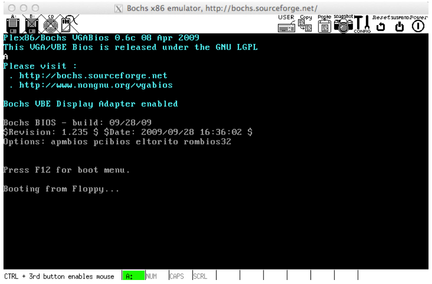
Today

- Project 1
- Processes

Project 1: Overview

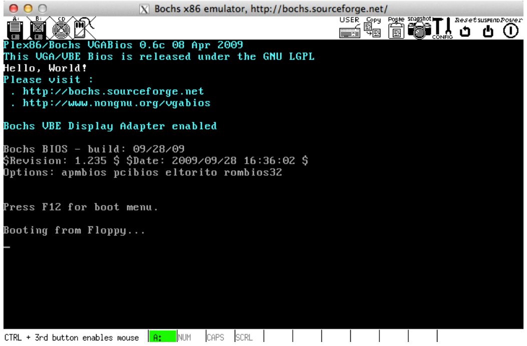
- Tools Set Up
  - Just bochs on the lab machines
  - Rest are installed for you
- Environment Set Up
  - Run source PathSetter.bash
- Displaying one character
  - Build, execution process
  - Bash script
- Displaying “Hello World”
- Adding functions to display

Screenshot: One Letter


Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: CD
USER Copy Paste snapshot CONFIG Reset SUSPEND Power
CTRL + 3rd button enables mouse
A: NUM CAPS SCRL

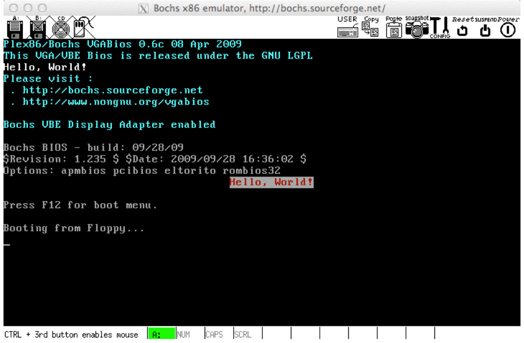
```
Plex86/Bochs VGABios 0.6c 08 Apr 2009
This VGA/VBE Bios is released under the GNU LGPL
A
Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs BIOS - build: 09/28/09
$Revision: 1.235 $ $Date: 2009/09/28 16:36:02 $
Options: apmbios pcibios eltorito rombios32


Press F12 for boot menu.

Booting from Floppy...
```

Screenshot: Hello, World!


Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: CD
USER Copy Paste snapshot CONFIG Reset SUSPEND Power
CTRL + 3rd button enables mouse
A: NUM CAPS SCRL

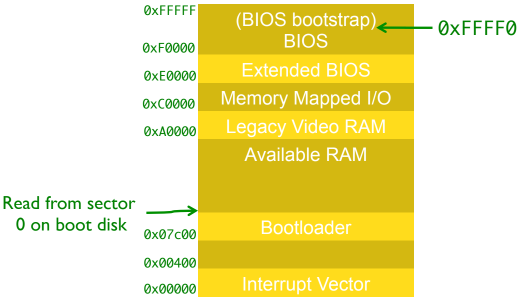
```
Plex86/Bochs VGABios 0.6c 08 Apr 2009
This VGA/VBE Bios is released under the GNU LGPL
Hello, World!
Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs BIOS - build: 09/28/09
$Revision: 1.235 $ $Date: 2009/09/28 16:36:02 $
Options: apmbios pcibios eltorito rombios32


Press F12 for boot menu.

Booting from Floppy...
_
```

Screenshot: Background Color


Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: CD
USER Copy Paste snapshot CONFIG Reset SUSPEND Power
CTRL + 3rd button enables mouse
A: NUM CAPS SCRL

```
Plex86/Bochs VGABios 0.6c 08 Apr 2009
This VGA/VBE Bios is released under the GNU LGPL
Hello, World!
Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs BIOS - build: 09/28/09
$Revision: 1.235 $ $Date: 2009/09/28 16:36:02 $
Options: apmbios pcibios eltorito rombios32
                                        Hello, World!

Press F12 for boot menu.

Booting from Floppy...
_
```

16-bit Real Mode Memory Map



16-bit Real Mode Registers

- General Purpose Registers:
  - ax, bx, cx, dx
    - Each holds 16 bits
  - Half registers:
    - ax = 0xABCD
      - ah = 0xAB al = 0xCD
      - ax = ah * 256 + al

Segmented Memory Access in 16-bit Real Mode

- Registers hold 16 bits but memory addresses are 20 bits?
- All addresses have 2 parts:
  - Segment – 16 bits
  - Offset – 16 bits
  - Ex: 0x1000 : 0xABCD
    segment offset
- Computing the actual address:
  - address = segment*0x10 + offset
- Add extra 0 to right of segment and add offset.
  - E.g. 0x1ABCD

Project 1: Hints

- Read through the directions
  - Later hints will help earlier parts
- Break up into small pieces
  - Just display one letter (as in example)
  - Then work on displaying a word
  - Then work on extensions
- Testing using gcc
  - Print out addresses in hex (%x)
    - Need to remove if using bcc

Project 1: Due next Wednesday

PROCESSES

What is a Process?

- Process – a sequential program execution
- Ideally, we would like our OS to be capable of running multiple processes/jobs at once (i.e., *multiprogramming*)
- **Challenge**: how to implement & ensure efficient use of system resources?

Difference between a process and a program

- Baking analogy:
  - Recipe = Program
  - Baker = Processor
  - Ingredients = data
  - Baking the cake = Process
- Interrupt analogy
  - The baker’s son runs in with a wounded hand
  - First aid guide = interrupt code

Main OS Process-related Goals

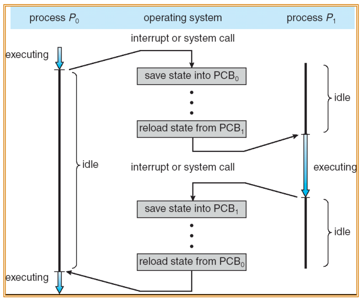
- Interleave the execution of existing processes to maximize processor utilization
- Provide reasonable response times
- Allocate resources to processes
- Support inter-process communication (and synchronization) and user creation of processes

Process Control Block (PCB)

- Used by OS/kernel to
  - Keep track of processes
  - switch between processes (i.e., context switch)
- Context switching wastes time (no new work), but enables multiprogramming

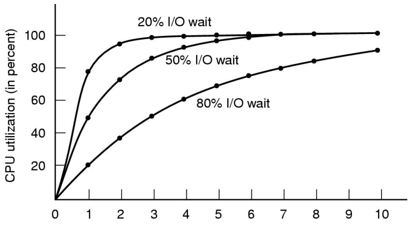
process state
process number
program counter
registers
memory limits
list of open files
• • •

CPU Switch From Process to Process



Modeling Multiprogramming

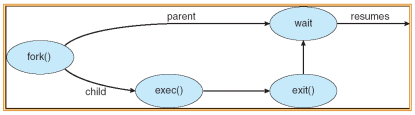
Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-**6006639**



CPU utilization as a function of the number of processes in memory.

PROCESSES IN UNIX

Process Creation



- Child is a complete copy of parent with a new id
- Exec() loads new executable image

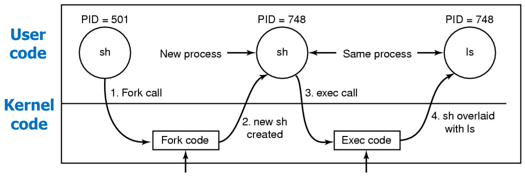
C Program Forking Separate Process

```
int main() {

   Pid_t  pid;

   pid = fork(); /* fork another process */

   if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        exit(-1);
   } else if (pid == 0) { /* child process */
        execlp("/bin/ls", "ls", NULL);
   } else { /* parent process */
   /* parent will wait for the child to complete */
        wait (NULL);
        printf ("Child %d Complete", pid);
        exit(0);
   }
}
```

Managing Processes (Unix)

- pid = fork() - create a child process
- wait(status) / waitpid(pid, status, opts)
  - wait for termination of a child. Either blocks, gets child return-code, or exit code (if no children)
- execvp(name, args)
  - replace image by name, with arguments args
  - Exec family
- exit(status)

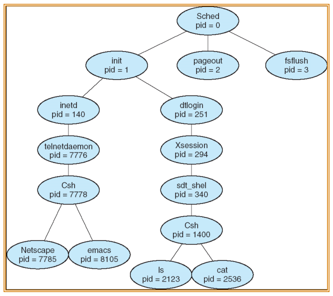
Executing the ls command



Allocate child's process table entry
Fill child's entry from parent
Allocate child's stack and user area
Fill child's user area from parent
Allocate PID for child
Set up child to share parent's text
Copy page tables for data and stack
Set up sharing of open files
Copy parent's registers to child

Find the executable program
Verify the execute permission
Read and verify the header
Copy arguments, environ to kernel
Free the old address space
Allocate new address space
Copy arguments, environ to stack
Reset signals
Initialize registers

Example tree of processes

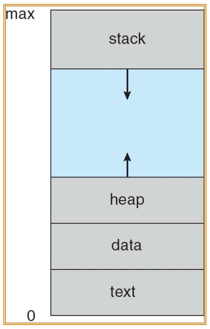


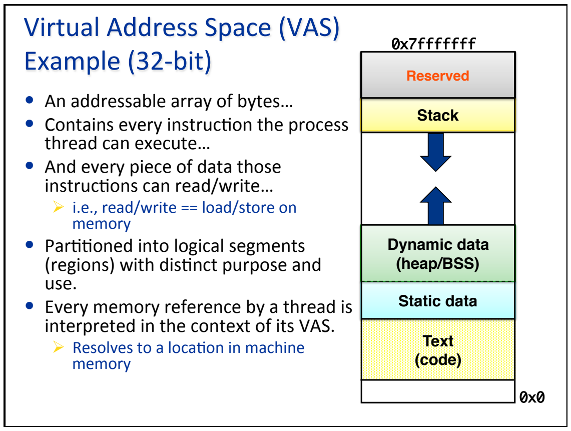
What does an OS need to do to allow multiprogramming?

- What resource concerns?

Process in Memory

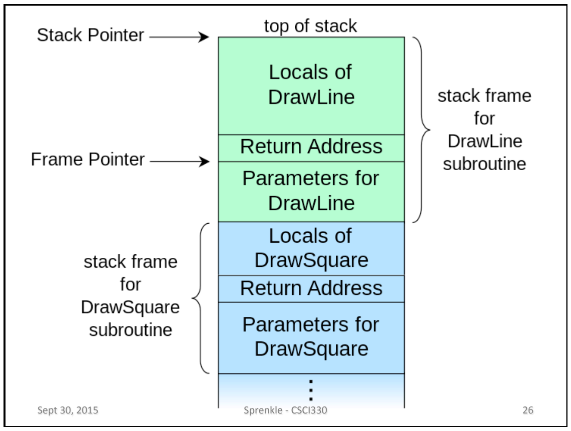
- A process includes:
  - **program counter**
    what line of program text to execute next
  - **stack**
    pointer to top of stack & frame pointer to calling function
  - **data section or heap**
    room to allocate data

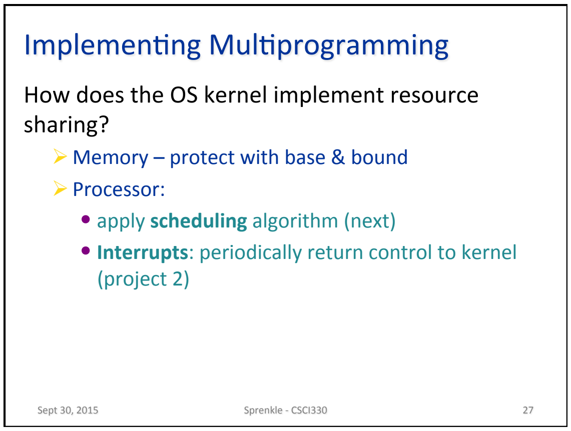




Virtual Address Space (VAS) Example (32-bit)

- An addressable array of bytes…
- Contains every instruction the process thread can execute…
- And every piece of data those instructions can read/write…
  - i.e., read/write == load/store on memory
- Partitioned into logical segments (regions) with distinct purpose and use.
- Every memory reference by a thread is interpreted in the context of its VAS.
  - Resolves to a location in machine memory

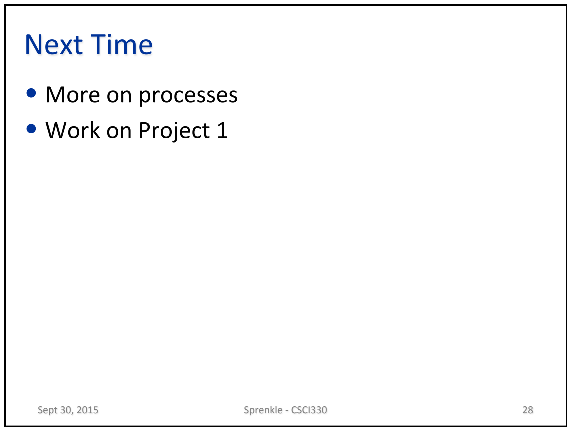




Implementing Multiprogramming

How does the OS kernel implement resource sharing?

- Memory – protect with base & bound
- Processor:
  - apply **scheduling** algorithm (next)
  - **Interrupts**: periodically return control to kernel (project 2)



Next Time

- More on processes
- Work on Project 1