Systems and Internet Infrastructure Security

Institute for Networking and Security Research
Department of Computer Science and Engineering
Pennsylvania State University, University Park, PA

Unix and Operating Systems (Part 1)

Devin J. Pohly <djpohly@cse.psu.edu>

To do:

- Download and install VirtualBox
- Download Xubuntu
  - Ubuntu with a more traditional interface
  - “i386” for 32-bit hardware, “amd64” for 64-bit
  - tiny.cc/311ubuntu for fast on-campus download
- Set up a virtual machine
  - Instructions will be posted on the website

Origins of Unix

- Developed in 1969 at Bell Labs
  - Intended as a development environment for multi-platform code
- Its use grew quickly, and the architectural advantages were embraced by the academic and industrial communities.
  - It dominates the "big iron" industrial environments
    - About 2/3 of servers run some Unix-like system (Dec. 2013)

Main attributes of Unix

- *multiuser* – supports multiple users on the system at the same time, each working with their own terminal
- *multitasking* – supports running multiple programs at a time
- *portable* – when changing hardware, only the lowest software layers need to be reimplemented

```
Memory: 21512k available (1200k kernel code, 452k data, 100k init, 0k highmem)
Calibrating delay loop... 484.96 BogoMIPS
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
checking if image is initramfs...it isn't (no cpio magic); looks like an initrd
Freeing initrd memory: 249k freed
POSIX conformance testing by UNIFIX
NET: Registered protocol family 16
pspfb: framebuffer at 0x04000000, mapped to 0x44000000, size 544k
pspfb: mode is 480x272x16, linelength=2048, pages=0
fb0: PSP frame buffer device
enable_irq(1) unbalanced
enable_irq(2) unbalanced
ikconfig 0.7 with /proc/config*
devfs: v1.22 (20021013) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x0
Console: switching to colour frame buffer device 80x34
pty: 256 Unix98 ptys configured
Generic RTC Driver v1.07
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
Console: switching to colour frame buffer device 80x34
Freeing unused kernel memory: 100k init
VFS: Cannot open root device "sda1" or 08:01
Please append a correct "root=" boot option
Kernel Panic: VFS: Unable to mount root fs on 08:01
```

Unix flavors over the years

- AT&T: the original Unix on research systems, 1969
  - Ken Thompson, Dennis Ritchie, Brian Kernighan, Doug McIlroy, et al.
  - System III, 1981
  - System V (SysV), 1983; System V Release 4 (SVR4), 1988–95
- University of California, Berkeley: BSD series, 1977–1995
  - Popular derivatives: FreeBSD, NetBSD, OpenBSD
  - Berkeley Software Design, Inc., 1991–2003, BSD/OS
- Microsoft: Xenix, 1980
  - Abandoned (“vaporware”), ported to 386 by SCO in 1987, succeeded by SCO OpenServer
- Sun Microsystems: SunOS, 1982 (later Solaris, OpenSolaris, acquired by Oracle in 2010)
- Hewlett-Packard: HP-UX, 1984
- Digital Equipment Corp. (DEC)/Compaq/HP, Ultrix, 1984 (later Digital Unix, Tru64 Unix)
- Carnegie-Mellon researchers: Mach (kernel), 1985–94
- IBM: AIX, 1986; z/OS (for mainframes), 2001
- Silicon Graphics Inc. (SGI): IRIX, 1988
- Open Software Foundation (DEC/IBM/HP/others): OSF/1, 1992 (based on Mach)
- Apple: Mac OS X, 2001 (derived from BSD and Mach)

Linux

- Linux kernel, 1991
  - Open-source Unix-like kernel that has seen broad adoption in both academia and industry
- GNU/Linux distributions
  - Red Hat, SUSE (Novell), Caldera (SCO, defunct), Debian, Mandrake/Mandriva, Slackware, Gentoo, Ubuntu, Knoppix, Fedora, Arch, and hundreds more
- Android, since 2003
  - Linux kernel
  - Open Handset Alliance, Android Open Source Project

Open source

- Many Unix-like systems in use today are distributed as “*open source*”
  - *Open source software is distributed with a license where the copyright allows the user of the source to review, modify, and distribute with no cost to anyone.*
  - Variants of this arrangement allow a person (a) to derive software from the distribution and charge or (b) never charge anyone for derivative works.
- Aside: “free” as in free beer (gratis) vs. free speech (libre)

LIVE FREE OR DIE
http://www.UNIX-systems.org
UNIX®
UNIX is a Registered Trademark of The Open Group

Unix architecture

- Software layers
  - *OS kernel* – direct interaction with hardware
  - *system calls* – interface to the kernel
  - *system libraries* – wrappers around system calls
  - *programming language libraries* – extend system libraries
  - *system utilities* – application-independent tools
    - fsck, fdisk, ip, mknod, mount, nfsd
  - *command interpreter* or *shell* – user interface
  - *application libraries* – application-specific tools
  - *applications* – complete programs for ordinary users
    - some applications have their own command shells and programming-language facilities (e.g., Perl, Python)

What's an OS?

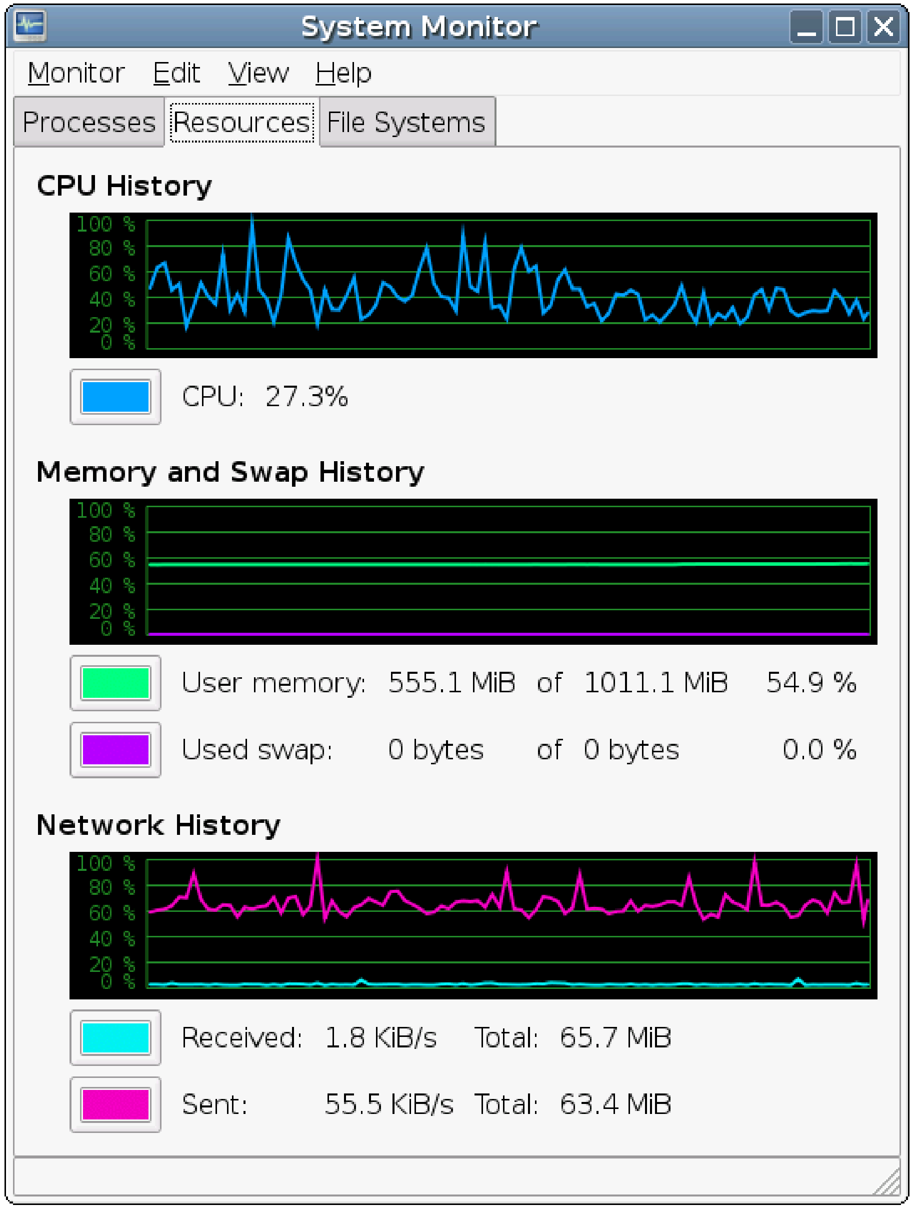
- Software that:
  - Directly interacts with hardware
    - OS is trusted to do so; user-level processes are not
    - OS must be ported to new HW; user-level programs are portable
  - Manages (allocates, schedules, protects) hardware resources
    - decides which programs can access which files, memory locations, pixels on the screen, etc., and when
  - Abstracts away messy hardware devices
    - provides high-level, convenient, portable abstractions
    - example: files vs. disk blocks
- Unix is a classic example of an OS.

OS as abstraction provider

- The OS is the "layer below"
  - a module that your program can call (with system calls)
  - a powerful API
    - Library functions in addition to system calls
    - Unix API is standardized as POSIX, from "portable operating system" + (Un)ix

OS as abstraction provider

- API supports useful abstractions
  - filesystem: open, read, write, close
  - network stack: connect, listen, read, write
  - virtual memory: brk, shm_open
  - process management: fork, wait, nice



CPU: 27.3%

User memory: 555.1 MiB of 1011.1 MiB 54.9 %

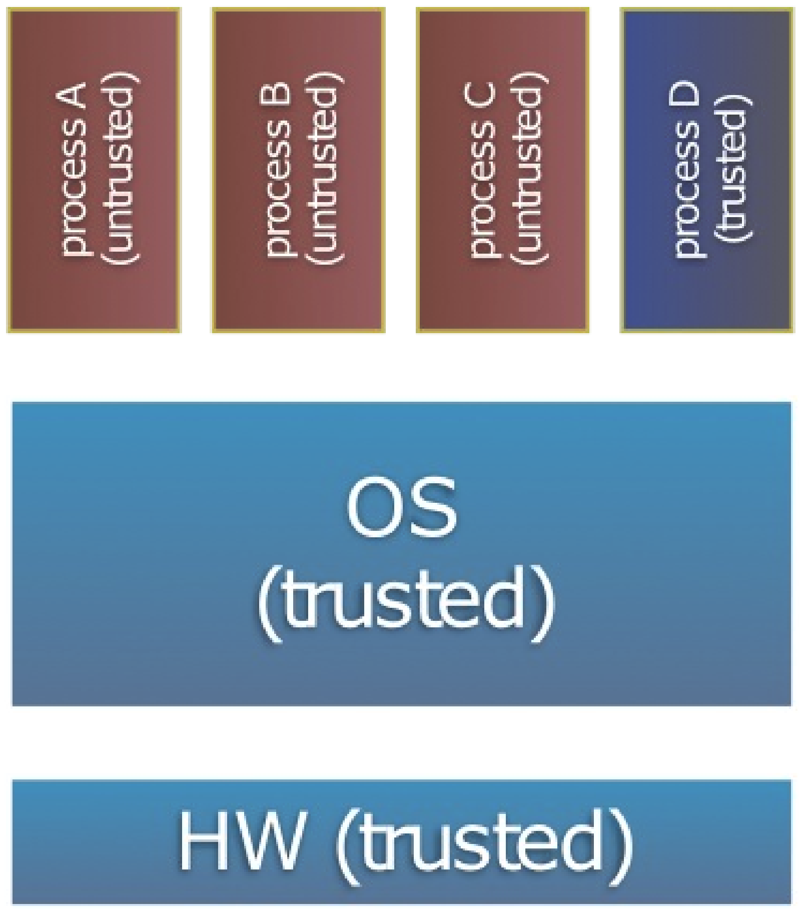
Used swap: 0 bytes of 0 bytes 0.0 %

Received: 1.8 KiB/s Total: 65.7 MiB

Sent: 55.5 KiB/s Total: 63.4 MiB

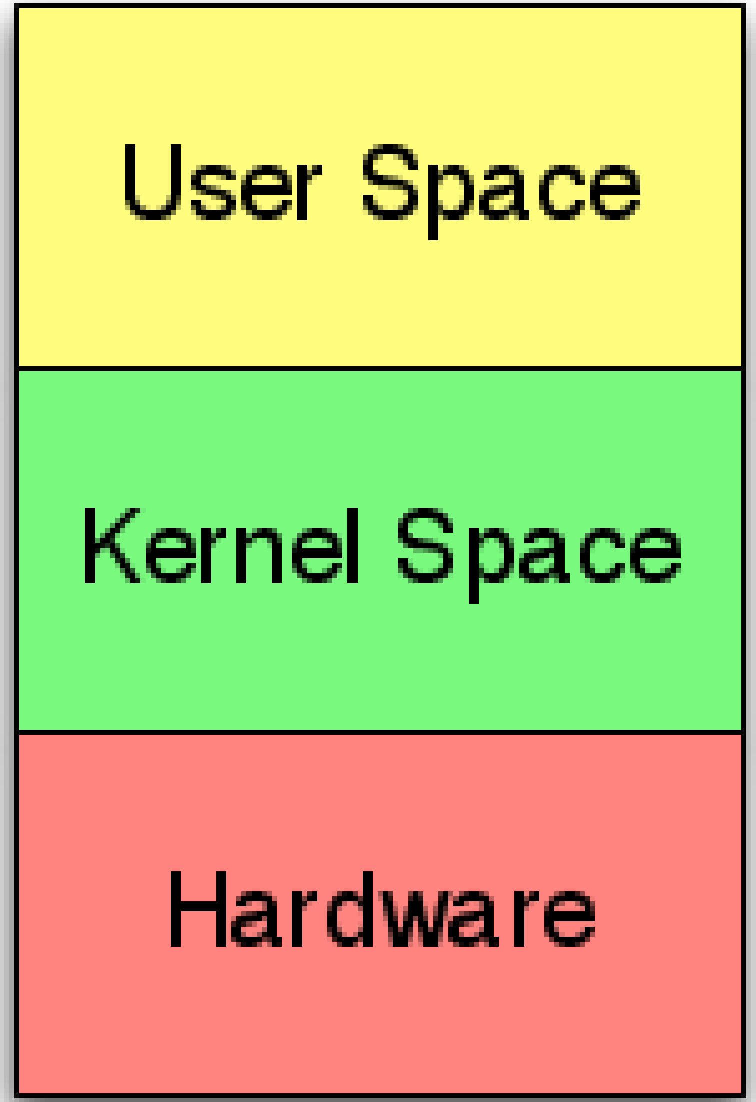
OS as protection system

- OS isolates processes from each other
  - permits controlled sharing via shared namespaces (e.g., filesystem names)
- OS isolates itself from processes
  - therefore must prevent processes from accessing hardware directly
- OS is allowed to access the hardware
  - user-level processes run with the CPU in an unprivileged mode
  - when the OS is running, the CPU is set to a privileged mode
  - user-level processes invoke a system call to safely enter the OS



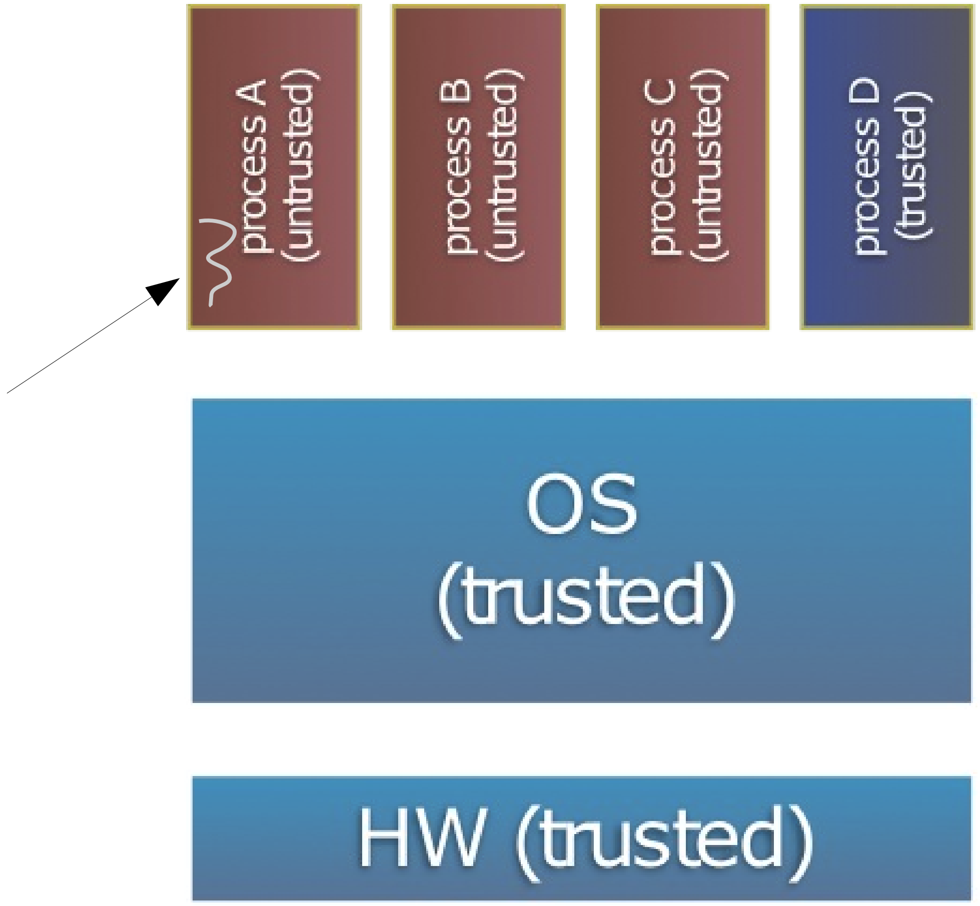
Hardware privilege modes

- A *privilege mode* is a CPU state that restricts the kinds of actions that code may perform
    - e.g., prevents direct access to hardware, process controls, and privileged/sensitive instructions
- There are two modes we are principally concerned about in this class: *user* and *kernel* modes
    - *user mode* has low privilege and is used for normal programs (and some system services) running in "userspace"
    - *kernel mode* has full privilege and is used by the operating system in "kernelspace"



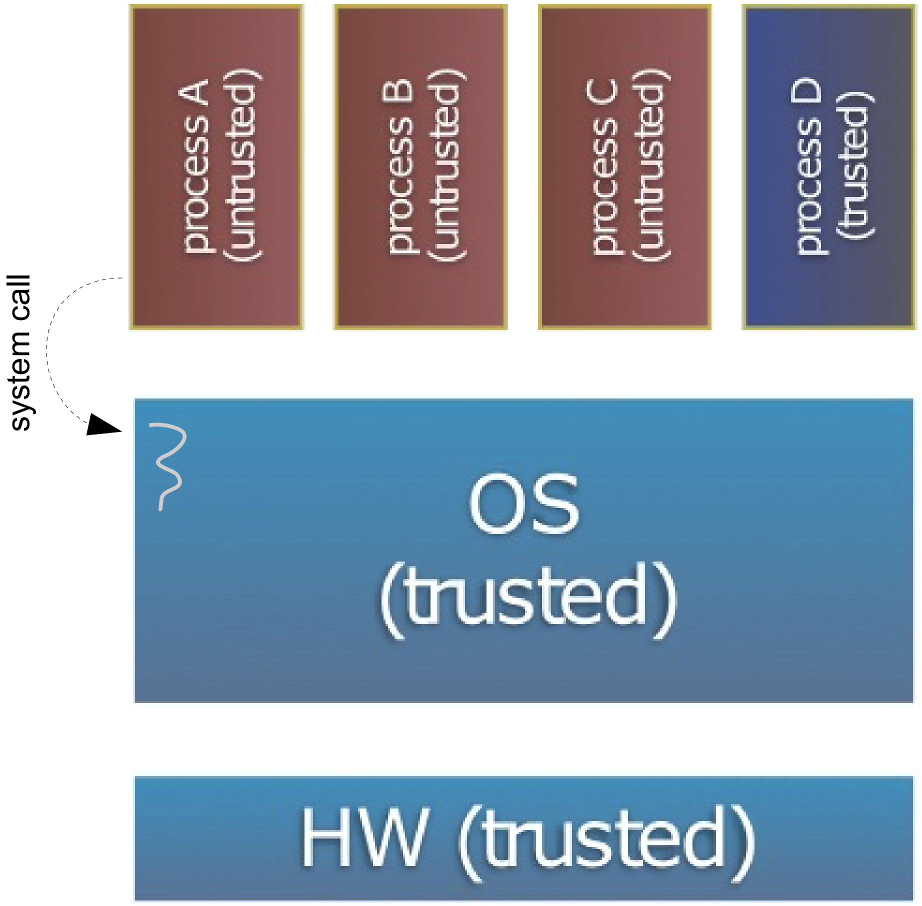
OS as protection system

A CPU (thread of execution) is running userspace code in process A; that CPU is set to *user mode*.



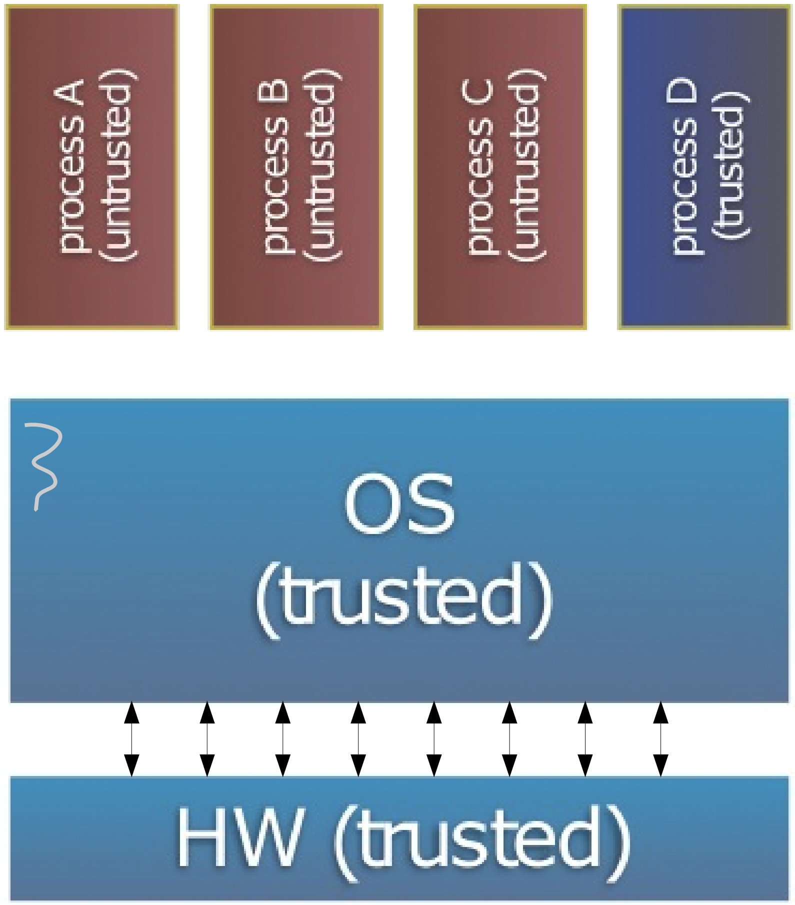
OS as protection system

Code in process A invokes a system call instruction or trap; the hardware then sets the CPU to *kernel mode* and gives control to the OS, which invokes the appropriate system call handler.



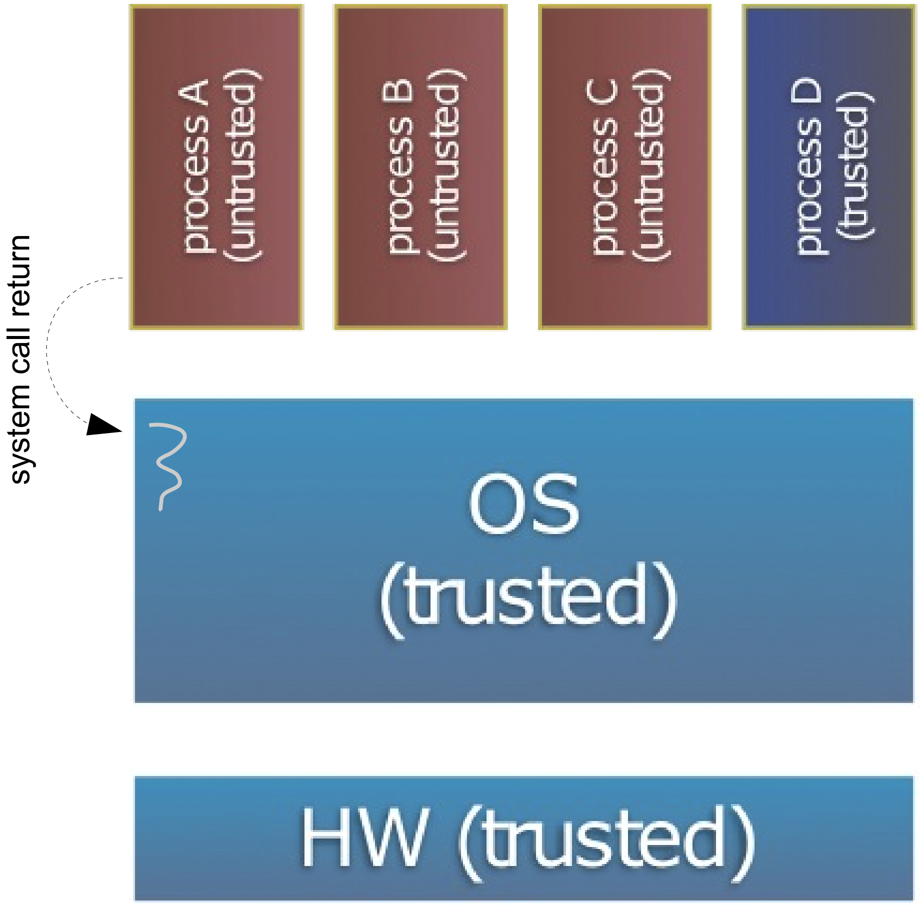
OS as protection system

Since the CPU executing the OS code is in kernel mode, it is able to use privileged instructions that interact directly with hardware devices such as disks.

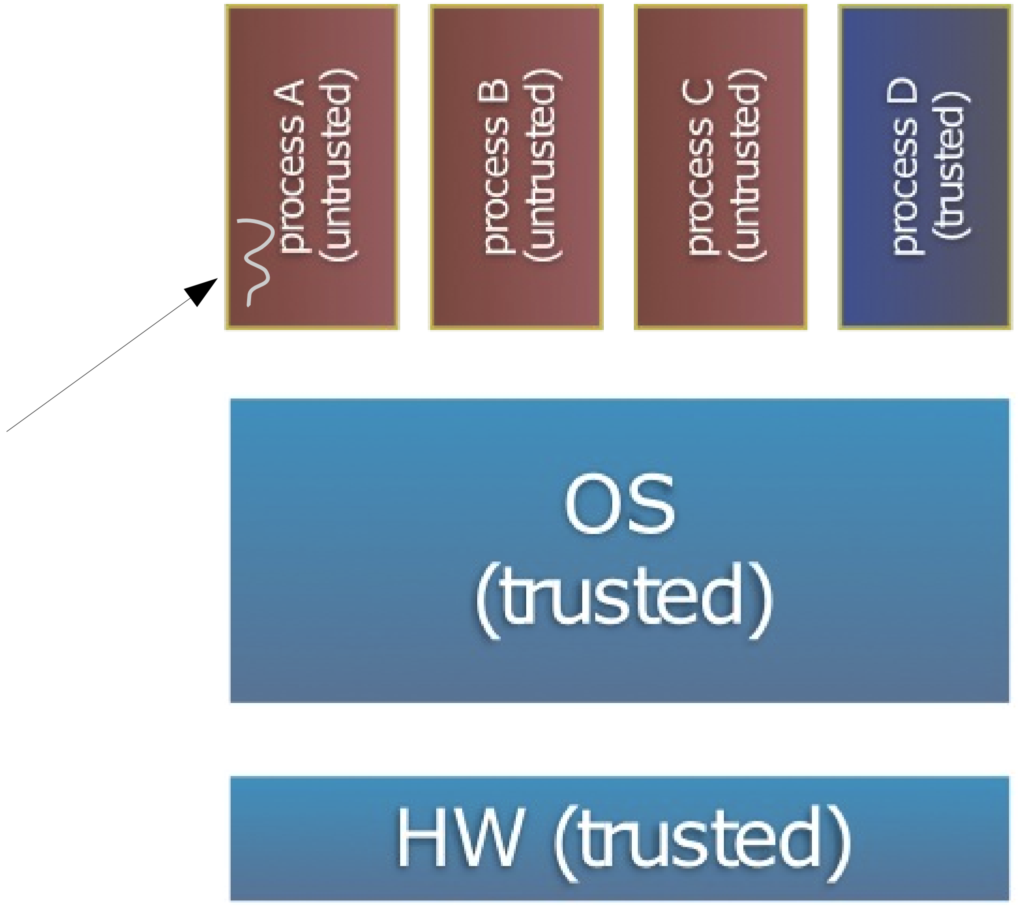


OS as protection system

Once the OS has finished handling the system call (which can involve long waits as it interacts with hardware), it sets the CPU back to user mode and returns control from the system call handler to the userspace code in process A.



OS as protection system



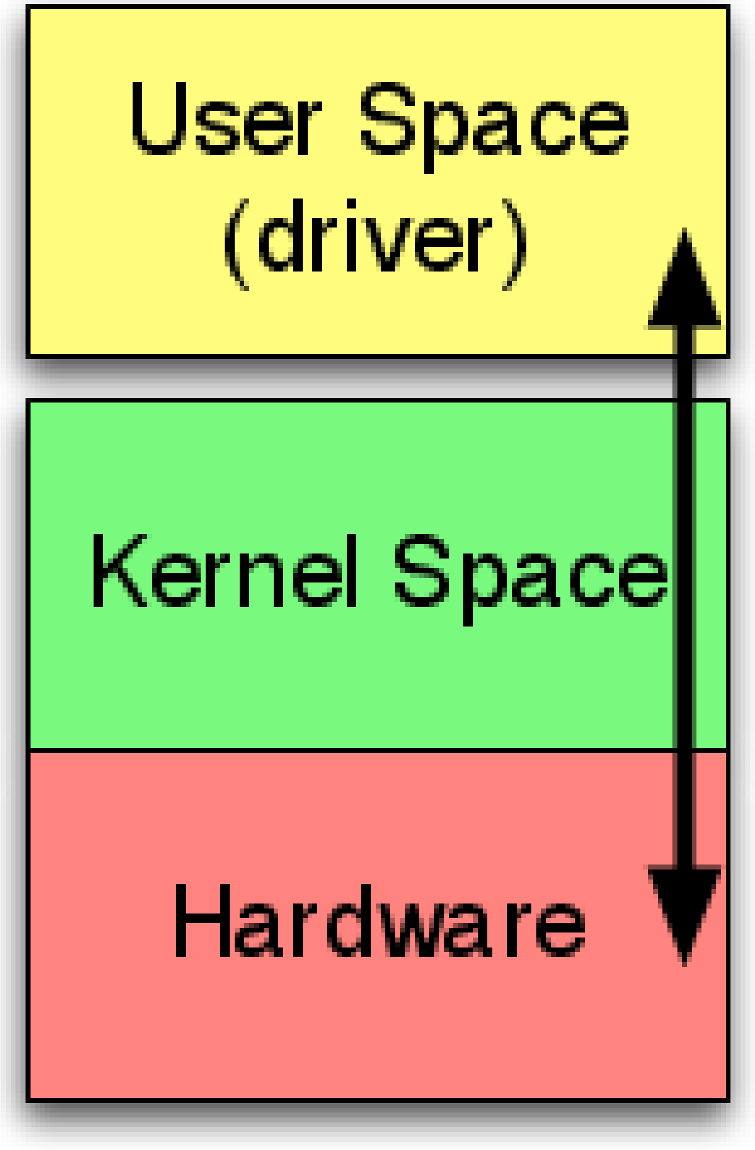
The process continues executing whatever code is next after the system call invocation.

Device drivers

- A *device driver* is a software module (program) that implements the interface to a piece of hardware (often needs kernel mode privilege)
  - e.g., printers, monitors, graphics cards, USB devices
  - often provided by the manufacturer of the device
  - for performance reasons, the driver is commonly run in kernelspace
  - device drivers were often compiled into the kernel
    - required the administrator to re-compile the operating system whenever support for a new device type was needed
    - each system had its own custom kernel

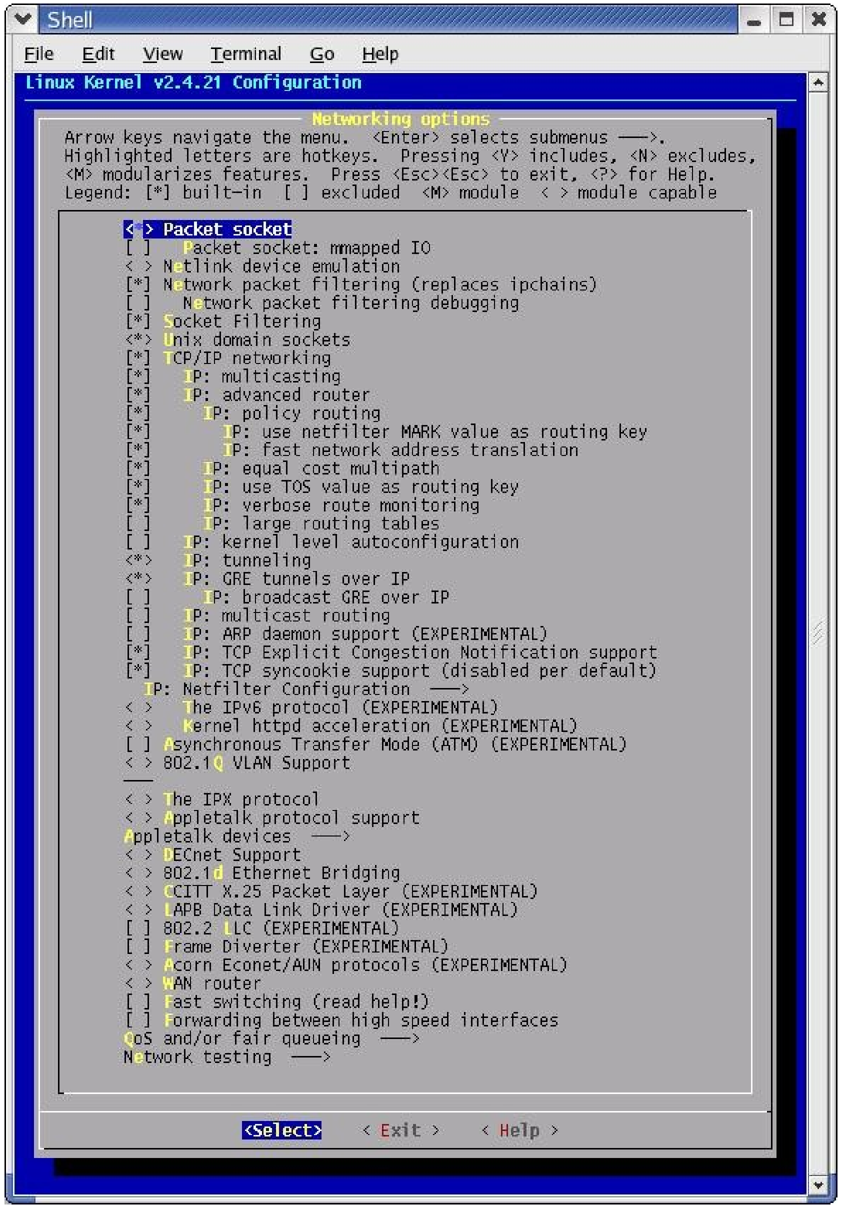
Recompiling kernels?

- Recompilation of the kernel is problematic
  - takes a long time
  - requires sophistication
  - versioning problems
- Solution 1: *userspace modules*
  - Userspace programs that support the operating system
    - leverages protection
    - allows independent patching and upgrading
    - removes dependency on kernel version (mostly)
    - Problem: performance. For high-speed hardware, context switching is costly.



Recompiling kernels?

- Solution 2: *kernel modules*
  - aka loadable kernel modules (LKM)
  - software modules that run in kernel space that can be loaded/unloaded on a running system
    - can extend the kernel functionality without recompilation
    - the trick is that the kernel provides generic interfaces (APIs) that the module uses to communicate with the kernel
    - used by almost every modern OS (OSX, Windows, Linux, etc.)
- PROTIP: To see what modules are loaded on your Unix system, use the "`lsmod`" command.



```
Linux Kernel v2.4.21 Configuration
                          Networking options
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        <*> Packet socket
        [ ]   Packet socket: mmapped IO
        < > Netlink device emulation
        [*] Network packet filtering (replaces ipchains)
        [ ]   Network packet filtering debugging
        [*] Socket Filtering
        <*> Unix domain sockets
        [*] TCP/IP networking
        [*]   IP: multicasting
        [*]   IP: advanced router
        [*]     IP: policy routing
        [*]       IP: use netfilter MARK value as routing key
        [*]       IP: fast network address translation
        [*]     IP: equal cost multipath
        [*]     IP: use TOS value as routing key
        [*]     IP: verbose route monitoring
        [ ]     IP: large routing tables
        [ ]   IP: kernel level autoconfiguration
        <*>   IP: tunneling
        <*>   IP: GRE tunnels over IP
        [ ]     IP: broadcast GRE over IP
        [ ]   IP: multicast routing
        [ ]   IP: ARP daemon support (EXPERIMENTAL)
        [*]   IP: TCP Explicit Congestion Notification support
        [*]   IP: TCP syncookie support (disabled per default)
          IP: Netfilter Configuration  --->
        < >   The IPv6 protocol (EXPERIMENTAL)
        < >   Kernel httpd acceleration (EXPERIMENTAL)
        [ ] Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)
        < > 802.1Q VLAN Support
        ---
        < > The IPX protocol
        < > Appletalk protocol support
        Appletalk devices  --->
        < > DECnet Support
        < > 802.1d Ethernet Bridging
        < > CCITT X.25 Packet Layer (EXPERIMENTAL)
        < > LAPB Data Link Driver (EXPERIMENTAL)
        [ ] 802.2 LLC (EXPERIMENTAL)
        [ ] Frame Diverter (EXPERIMENTAL)
        < > Acorn Econet/AUN protocols (EXPERIMENTAL)
        < > WAN router
        [ ] Fast switching (read help!)
        [ ] Forwarding between high speed interfaces
        QoS and/or fair queueing  --->
        Network testing  --->

                 <Select>    < Exit >    < Help >
```