

**School of Computer Science and
Engineering
Bharathidasan University**

OPEN SOURCE TECHNOLOGIES

UNIT-1

Dr. M. Durairaj
Associate Professor
School of Computer Science,
Engineering and Applications
Bharathidasan University

Open Source Software - Definition

“OSS is licensed software in which the source code is made available to users to enable them to modify it for their own purposes and (within certain restrictions) redistribute original and derived works as they see fit.”

- No one has exclusive control over the term “open source”
- Not an enforceable copyrighted term or trademark
- Open Source Initiative (OSI) www.opensource.org – was founded in 1998 & has unofficial power over the core concepts

Open Source Software – Definition

Free redistribution	“License shall not require a royalty or other fee for such sale”
Source code	Must include source code & allow distribution (or a well-publicized means of obtaining the source code)
Derived works	Must allow modifications & allow them to be distributed
Integrity of author’s source code	License must permit distribution of software built from modified source code
No discrimination against	Persons, groups or fields of endeavor (e.g. genetic research)
Distribution of license	Rights to program must apply to all without the need for execution of additional license
License must not be specific to a product	The rights attached to a program must not depend on the program’s being part of a particular software distribution
License must not restrict other software	Must not insist all other programs distributed on the same medium must be open-source software
License must be technology-neutral	No provision of the license may be predicated on any individual technology or style of interface

Source: <http://opensource.org/docs/definition.php>; viewed 4/13/09

Free Software

- “Free” software “is [software](#) that can be used, studied, and modified,” copied, changed with little or no restriction, and which can be copied and redistributed in modified or unmodified form. Free software is available gratis (free of charge) in most cases.
- “In practice, for software to be distributed as free software, the human-readable form of the program (the [source code](#)) must be made available” along “with a notice granting the” user permission to further adapt the code and continue its redistribution for free.
- This notice either grants a "[free software license](#)", or releases the source code into the [public domain](#).

Open Source Software

- In the beginning, all software was free
 - in the 1960s ,when IBM and others sold the first large-scale computers, these machines came with software which was free.
 - This software could be freely shared among users,
 - The software came written in a programming language (source code available), and it could be improved and modified.
 - Manufacturers were happy that people were writing software that made their machines useful. ⁽¹⁾
- Then proprietary software dominated the software landscape as manufacturers removed access to the source code.
 - IBM and others realized that most users couldn't or didn't want to "fix" their own software and
 - There was money to be made in leasing or licensing software.

Open Source Software

- By the mid-1970s almost all software was proprietary
 - **“*Proprietary software* is software that is owned by an individual or a company (usually the one that developed it). There are almost always major restrictions on its use, and its source code is almost always kept secret.”** ⁽¹⁾ users were not allowed to redistribute it,
 - source code is not available
 - users cannot modify the programs.
 - Software is an additional product that was for sale
 - In 1980 US copyright law was modified to include software
- ⁽¹⁾

Open Source Software

- In late 1970s and early 1980s, two different groups started what became known as the open source software movement:
- East coast, Richard Stallman (1985), formerly a programmer at the MIT AI Lab, launched the GNU Project and the Free Software Foundation.
 - “to satisfy the need for and give the benefit of ‘software freedom’ to computer users.” (1)
 - ultimate goal of the GNU Project was to build a free operating system
 - the GNU General Public License (GPL) was designed to ensure that the software produced by GNU will remain free, and to promote the production of more and more free software.

Free Software Foundation

Guiding Principles

- “Free software is a matter of liberty, not price.
- To understand the concept, you should think of free as in **free speech (right)**, not as in **free beer (gift)**.
- Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software.
 - The freedom to run the program, for any purpose (freedom 0).
 - The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
 - The freedom to redistribute copies so you can help your neighbor (freedom 2).
 - The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.” <http://www.gnu.org/philosophy/free-sw.html>

Free Software Foundation

Guiding Principles

- “A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to [anyone anywhere](#). Being free to do these things means (among other things) that you do not have to ask or pay for permission.
- You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be required to notify anyone in particular, or in any particular way.” <http://www.gnu.org/philosophy/free-sw.html>
- This is a philosophy, a world view.

Free Software Foundation

- Very counter-culture
- **Hacker** is considered a “good-guy”
 - “**Hacker (computer security)** someone involved in computer security/insecurity
 - **Hacker (programmer subculture)**, a programmer subculture originating in the US academia in the 1960s, which is nowadays mainly notable for the free software/open source movement
 - **Hacker (hobbyist)**, an enthusiastic home computer hobbyist”
<http://en.wikipedia.org/wiki/Hacker>
- **Cracker** is a “bad-guy”
 - A **cracker** is someone who cracks software or digital media
 - “**Software cracking** is the modification of software to remove protection methods: copy protections, trial/demo version, serial number, hardware key, date checks, CD check or software annoyances like nag screens and adware”. [http://en.wikipedia.org/wiki/Cracker_\(computing\)](http://en.wikipedia.org/wiki/Cracker_(computing))

Open Source Software

- West coast, the Computer Science Research Group (CSRG) of the University of California at Berkeley was improving the Unix system, and building applications which quickly become “BSD Unix”.
- Unix was initially developed by AT&T employees ⁽¹⁾
 - efforts were funded mainly by DARPA contracts
 - a network of Unix programmers around the world helped to debug, maintain and improve the system.
 - in late 1980s, distributed under the “BSD license” (one of the first open source licenses).
 - Unfortunately, still contained some components that were proprietary requiring a license from AT&T

Adapted from http://eu.conecta.it/paper/brief_history_open_source.html

(1)<http://en.wikipedia.org/wiki/UNIX>

Open Source Software

- During the 1980s and early 1990s, open source software continued its development, initially in several relatively isolated groups.
- Slowly, much of the software was integrated
- The various groups merged
- As a result of this, complete operating environments could be built on top of Unix using open source software.
- Many Internet ISPs use UNIX as their operating system of choice.

Open Source Software

- 1991-1992, the open source world improved
- In California, Bill Jolitz implementing a version of BSD Unix free of AT & T's copyright.
 - The work was covered by the BSD license making it completely free.
 - It included other free software GNU licenses

Open Source Software

- Also during 1991-1992
- In Finland, Linus Torvalds, a Finnish computer science student, was implementing the first versions of Linux.
- Other people joined to collaboration to create the GNU/Linux operating system.
- By 1993, both GNU/Linux and BSD Unix were free stable operating environments.
 - Both continue to evolve

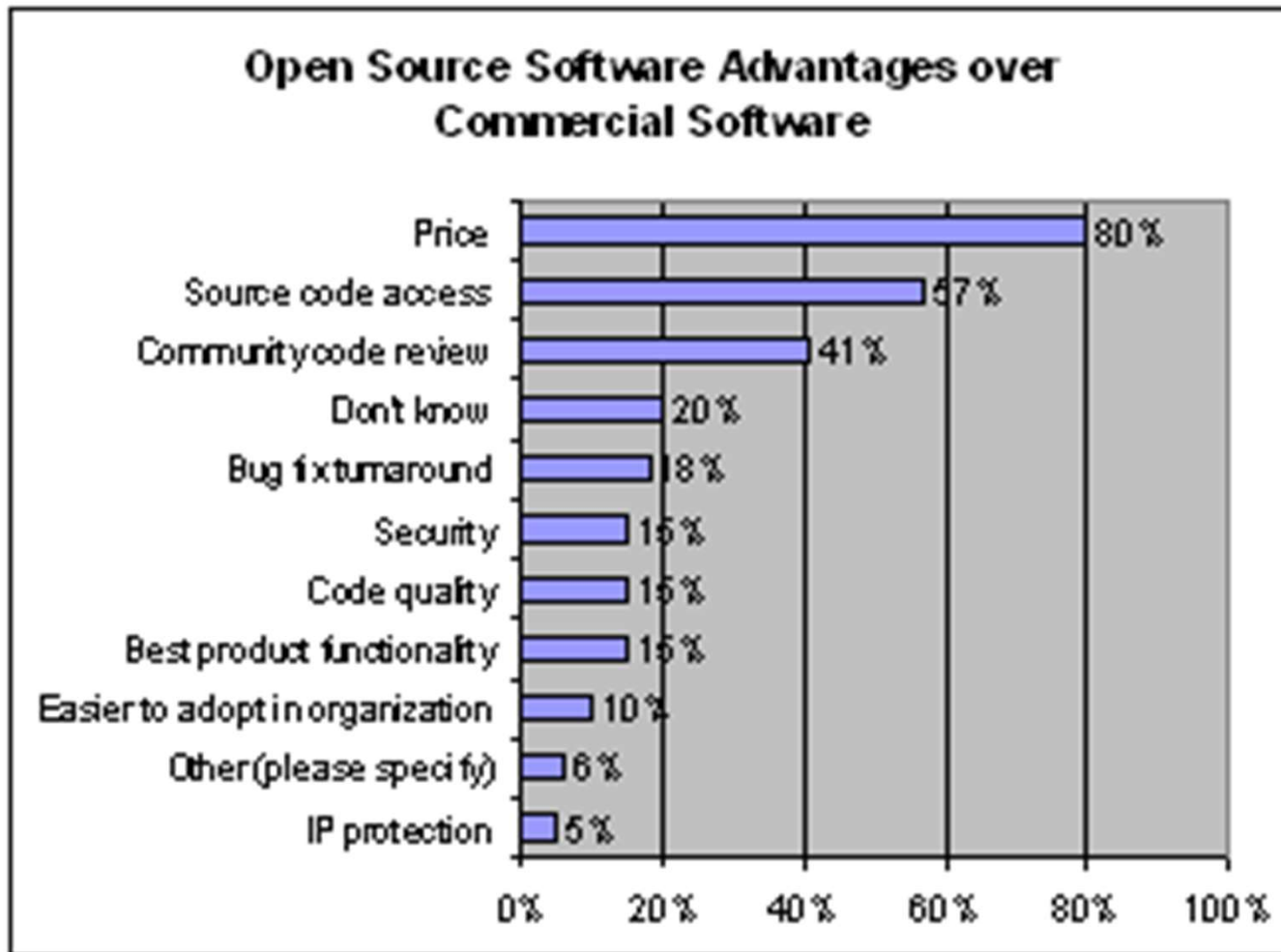
Open Source Software

- “Open source is a development method for software that harnesses the power of distributed peer review and transparency of process.
- The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.
- The Open Source Initiative (OSI) is a non-profit corporation formed to educate about and advocate for the benefits of open source.”
- OSI includes a standards body, maintaining the Open Source Definition for the good of the community.

Open Source Software

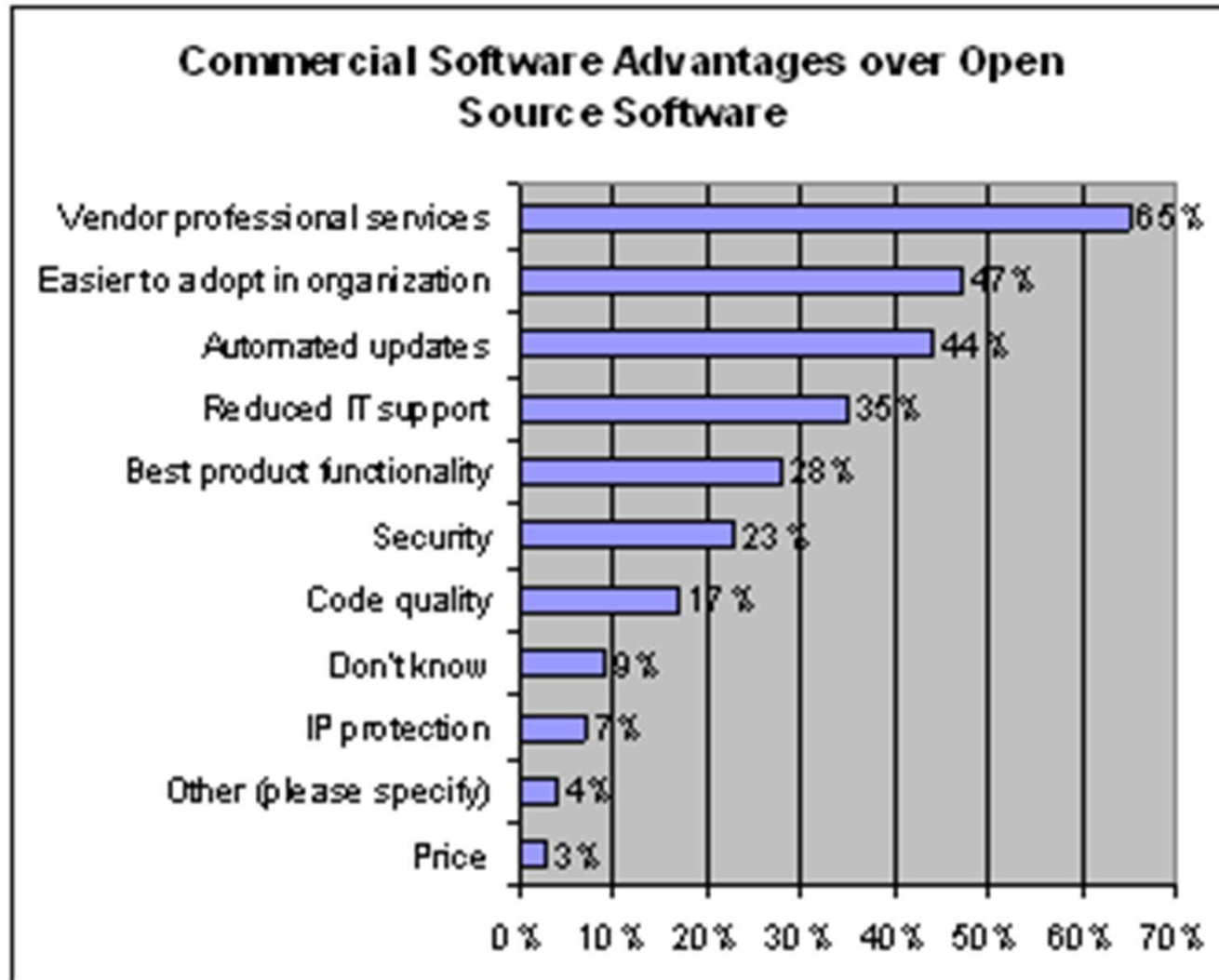
- Today there are many who believe proprietary software is the only possible model
 - Microsoft
 - Apple, especially for the iPod and iPhone
- Recently the software industry has begun to consider free software as an option again.
 - Apple's OS X and Leopard are based on Unix
 - Google's Chrome
 - Mozilla Firefox

Open Source vs Proprietary Software



Why choose proprietary software over open source? Survey says! by Matt Asay http://news.cnet.com/8301-13505_3-9789275-16.html

Open Source vs Proprietary Software



Why choose proprietary software over open source? Survey says! by Matt Asay http://news.cnet.com/8301-13505_3-9789275-16.html

Open Source Software

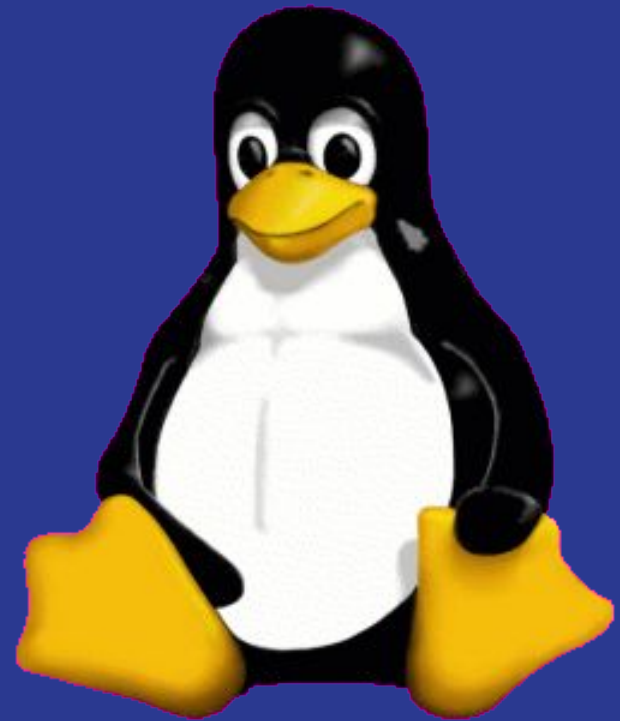
See for Yourself

Revolution OS

- “Revolution OS is a 2001 documentary which traces the history of GNU, Linux, and the open source and free software movements. “
<http://video.google.ca/videoplay?docid=7707585592627775409>
- Richard Stallman & Opensource
http://www.youtube.com/watch?v=kSZZraHN0Yg&feature=PlayList&p=65CA10D0F42E48FD&playnext=1&playnext_from=PL&index=7
- Free software CNet
http://www.youtube.com/watch?v=a9fqll9B6QU&feature=PlayList&p=65CA10D0F42E48FD&playnext=1&playnext_from=PL&index=9
- Linux -- IBM
<http://www.youtube.com/watch?v=KwEWxpOWOok>
- Linux commercials
<http://www.youtube.com/watch?v=aufL76bXLAg&feature=related>

The Linux Kernel: Introduction

Dr. M. Durairaj
Associate Professor
School of Computer Science,
Engineering and Applications
Bharathidasan University



History

- UNIX: 1969 Thompson & Ritchie AT&T Bell Labs.
- BSD: 1978 Berkeley Software Distribution.
- Commercial Vendors: Sun, HP, IBM, SGI, DEC.
- GNU: 1984 Richard Stallman, FSF.
- POSIX: 1986 IEEE Portable Operating System unIX.
- Minix: 1987 Andy Tannenbaum.
- SVR4: 1989 AT&T and Sun.
- Linux: 1991 Linus Torvalds Intel 386 (i386).
- Open Source: GPL.

Linux Features

- UNIX-like operating system.
- Features:
 - Preemptive multitasking.
 - Virtual memory (protected memory, paging).
 - Shared libraries.
 - Demand loading, dynamic kernel modules.
 - Shared copy-on-write executables.
 - TCP/IP networking.
 - SMP support.
 - Open source.

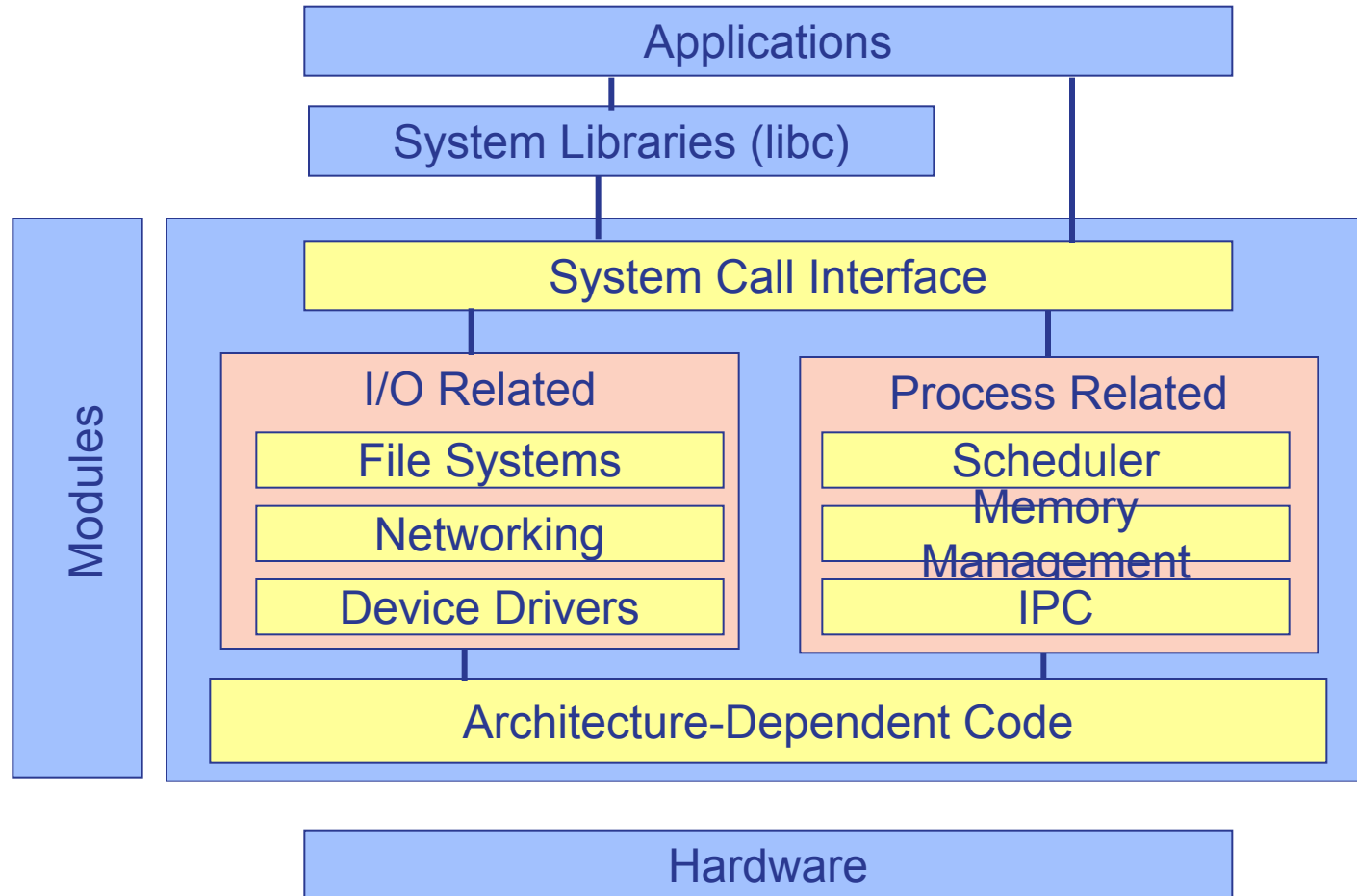
What's a Kernel?

- AKA: executive, system monitor.
- Controls and mediates access to hardware.
- Implements and supports fundamental abstractions:
 - Processes, files, devices etc.
- Schedules / allocates system resources:
 - Memory, CPU, disk, descriptors, etc.
- Enforces security and protection.
- Responds to user requests for service (system calls).
- Etc...etc...

Kernel Design Goals

- Performance: efficiency, speed.
 - Utilize resources to capacity with low overhead.
- Stability: robustness, resilience.
 - Uptime, graceful degradation.
- Capability: features, flexibility, compatibility.
- Security, protection.
 - Protect users from each other & system from bad users.
- Portability.
- Extensibility.

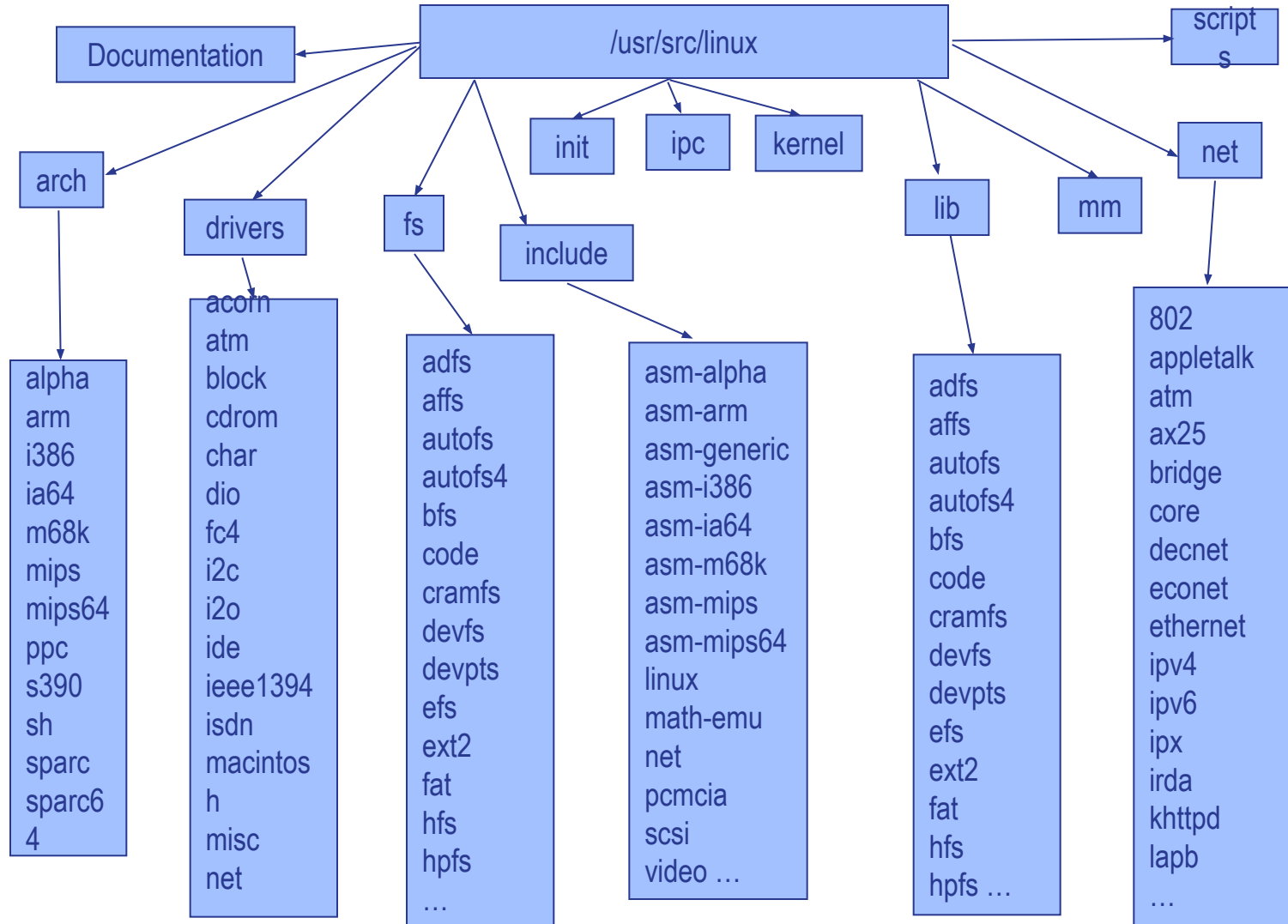
Example “Core” Kernel



Architectural Approaches

- Monolithic.
- Layered.
- Modularized.
- Micro-kernel.
- Virtual machine.

Linux Source Tree Layout



linux/arch

- Subdirectories for each current port.
- Each contains **kernel**, **lib**, **mm**, **boot** and other directories whose contents override code stubs in architecture independent code.
- **lib** contains highly-optimized common utility routines such as memcpy, checksums, etc.
- **arch** as of 2.4:
 - alpha, arm, i386, ia64, m68k, mips, mips64.
 - ppc, s390, sh, sparc, sparc64.

linux/drivers

- Largest amount of code in the kernel tree (~1.5M).
- device, bus, platform and general directories.
- drivers/char – n_tty.c is the default line discipline.
- drivers/block – elevator.c, genhd.c, linear.c, ll_rw_blk.c, raidN.c.
- drivers/net – specific drivers and general routines Space.c and net_init.c.
- drivers/scsi – scsi_*.c files are generic; sd.c (disk), sr.c (CD-ROM), st.c (tape), sg.c (generic).
- General:
 - cdrom, ide, isdn, parport, pcmcia, pnp, sound, telephony, video.
- Buses – fc4, i2c, nubus, pci, sbus, tc, usb.
- Platforms – acorn, macintosh, s390, sgi.

linux/fs

- Contains:
 - virtual filesystem (VFS) framework.
 - subdirectories for actual filesystems.
- vfs-related files:
 - exec.c, binfmt_*.c - files for mapping new process images.
 - devices.c, blk_dev.c – device registration, block device support.
 - super.c, filesystems.c.
 - inode.c, dcache.c, namei.c, buffer.c, file_table.c.
 - open.c, read_write.c, select.c, pipe.c, fifo.c.
 - fcntl.c, ioctl.c, locks.c, dquot.c, stat.c.

linux/include

- include/asm-*:
 - Architecture-dependent include subdirectories.
- include/linux:
 - Header info needed both by the kernel and user apps.
 - Usually linked to /usr/include/linux.
 - Kernel-only portions guarded by #ifdefs
 - #ifdef __KERNEL__
 - /* kernel stuff */
 - #endif
- Other directories:
 - math-emu, net, pcmcia, scsi, video.

linux/init

- Just two files: version.c, main.c.
- version.c – contains the version banner that prints at boot.
- main.c – architecture-independent boot code.
- start_kernel is the primary entry point.

linux/ipc

- System V IPC facilities.
- If disabled at compile-time, util.c exports stubs that simply return `-ENOSYS`.
- One file for each facility:
 - `sem.c` – semaphores.
 - `shm.c` – shared memory.
 - `msg.c` – message queues.

linux/kernel

- The core kernel code.
- sched.c – “the main kernel file”:
 - scheduler, wait queues, timers, alarms, task queues.
- Process control:
 - fork.c, exec.c, signal.c, exit.c etc...
- Kernel module support:
 - kmod.c, ksyms.c, module.c.
- Other operations:
 - time.c, resource.c, dma.c, softirq.c, itimer.c.
 - printk.c, info.c, panic.c, sysctl.c, sys.c.

linux/lib

- kernel code cannot call standard C library routines.
- Files:
 - `brlock.c` – “Big Reader” spinlocks.
 - `cmdline.c` – kernel command line parsing routines.
 - `errno.c` – global definition of `errno`.
 - `inflate.c` – “gunzip” part of `gzip.c` used during boot.
 - `string.c` – portable string code.
 - Usually replaced by optimized, architecture-dependent routines.
 - `vsprintf.c` – `libc` replacement.

linux/mm

- Paging and swapping:
 - swap.c, swapfile.c (paging devices), swap_state.c (cache).
 - vmscan.c – paging policies, kswapd.
 - page_io.c – low-level page transfer.
- Allocation and deallocation:
 - slab.c – slab allocator.
 - page_alloc.c – page-based allocator.
 - vmalloc.c – kernel virtual-memory allocator.
- Memory mapping:
 - memory.c – paging, fault-handling, page table code.
 - filemap.c – file mapping.
 - mmap.c, mremap.c, mlock.c, mprotect.c.

linux/scripts

- Scripts for:
 - Menu-based kernel configuration.
 - Kernel patching.
 - Generating kernel documentation.

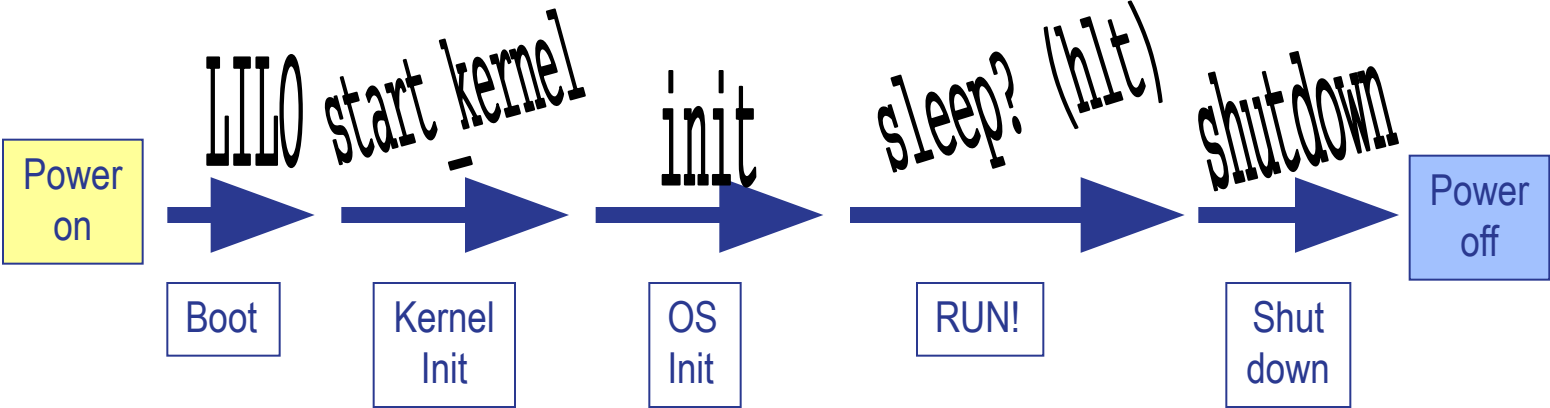
Summary

- Linux is a modular, UNIX-like monolithic kernel.
- Kernel is the heart of the OS that executes with special hardware permission (kernel mode).
- “Core kernel” provides framework, data structures, support for drivers, modules, subsystems.
- Architecture dependent source sub-trees live in /arch.

Booting and Kernel Initialization



System Lifecycle: Ups & Downs



Boot Terminology



- Loader:
 - Program that moves bits from disk (usually) to memory and then transfers CPU control to the newly “loaded” bits (executable).
- Bootloader / Bootstrap:
 - Program that loads the “first program” (the kernel).
- Boot PROM / PROM Monitor / BIOS:
 - Persistent code that is “already loaded” on power-up.
- Boot Manager:
 - Program that lets you choose the “first program” to load.

LILO: Linux LOader

- A versatile boot manager that supports:
 - Choice of Linux kernels.
 - Boot time kernel parameters.
 - Booting non-Linux kernels.
 - A variety of configurations.
- Characteristics:
 - Lives in MBR or partition boot sector.
 - Has no knowledge of filesystem structure so...
 - Builds a sector “map file” (block map) to find kernel.
- `/sbin/lilo` – “map installer”.
 - `/etc/lilo.conf` is lilo configuration file.



Example lilo.conf File

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux

image=/boot/vmlinuz-2.2.12-20
    label=linux
    initrd=/boot/initrd-2.2.12-20.img
    read-only
    root=/dev/hda1
```

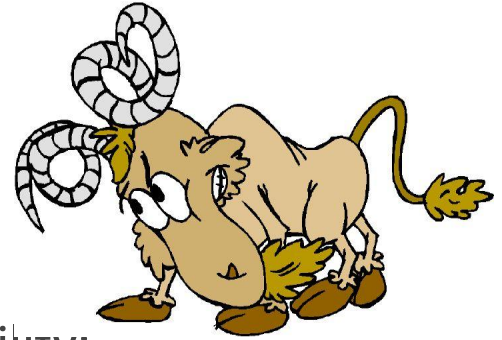
/sbin/init

- Ancestor of all processes (except idle/swapper process).
- Controls transitions between “runlevels”:
 - 0: shutdown
 - 1: single-user
 - 2: multi-user (no NFS)
 - 3: full multi-user
 - 5: X11
 - 6: reboot
- Executes startup/shutdown scripts for each runlevel.

Shutdown

- Use `/bin/shutdown` to avoid data loss and filesystem corruption.
- Shutdown inhibits login, asks init to send SIGTERM to all processes, then SIGKILL.
- Low-level commands: `halt`, `reboot`, `poweroff`.
 - Use `-h`, `-r` or `-p` options to shutdown instead.
- Ctrl-Alt-Delete “Vulcan neck pinch”:
 - defined by a line in `/etc/inittab`.
 - `ca::ctrlaltdel:/sbin/shutdown -t3 -r now.`





Advanced Boot Concepts

- Initial ramdisk (initrd) – two-stage boot for flexibility:
 - First mount “initial” ramdisk as root.
 - Execute linuxrc to perform additional setup, configuration.
 - Finally mount “real” root and continue.
 - See Documentation/initrd.txt for details.
 - Also see “man initrd”.
- Net booting:
 - Remote root (Diskless-root-HOWTO).
 - Diskless boot (Diskless-HOWTO).

Summary

- Bootstrapping a system is a complex, device-dependent process that involves transition from hardware, to firmware, to software.
- Booting within the constraints of the Intel architecture is especially complex and usually involves firmware support (BIOS) and a boot manager (LILO).
- `/sbin/lilo` is a “map installer” that reads configuration information and writes a boot sector and block map files used during boot.
- `start_kernel` is Linux “main” and sets up process context before spawning process 0 (idle) and process 1 (init).
- The `init()` function performs high-level initialization before exec'ing the user-level init process.

System Calls



System Calls

- Interface between user-level processes and hardware devices.
 - CPU, memory, disks etc.
- Make programming easier:
 - Let kernel take care of hardware-specific issues.
- Increase system security:
 - Let kernel check requested service via syscall.
- Provide portability:
 - Maintain interface but change functional implementation.

POSIX APIs

- API = Application Programmer Interface.
 - Function defn specifying how to obtain service.
 - By contrast, a system call is an explicit request to kernel made via a software interrupt.
- Standard C library (**libc**) contains wrapper routines that make system calls.
 - e.g., malloc, free are libc routines that use the brk system call.
- POSIX-compliant = having a standard set of APIs.
- Non-UNIX systems can be POSIX-compliant if they offer the required set of APIs.

Linux System Calls (1)

Invoked by executing `int $0x80`.

- Programmed exception vector number 128.
- CPU switches to kernel mode & executes a kernel function.
- Calling process passes **syscall number** identifying system call in **eax** register (on Intel processors).
- Syscall handler responsible for:
 - Saving registers on kernel mode stack.
 - Invoking syscall service routine.
 - Exiting by calling `ret_from_sys_call()`.

Linux System Calls (2)

- System call dispatch table:
 - Associates syscall number with corresponding service routine.
 - Stored in `sys_call_table` array having up to `NR_syscall` entries (usually 256 maximum).
 - nth entry contains service routine address of syscall n.

Initializing System Calls

- `trap_init()` called during kernel initialization sets up the IDT (interrupt descriptor table) entry corresponding to vector 128:
 - `set_system_gate(0x80, &system_call);`
- A system gate descriptor is placed in the IDT, identifying address of `system_call` routine.
 - Does not disable maskable interrupts.
 - Sets the descriptor privilege level (DPL) to 3:
 - Allows User Mode processes to invoke exception handlers (i.e. syscall routines).

The `system_call()` Function

- Saves syscall number & CPU registers used by exception handler on the stack, except those automatically saved by control unit.
- Checks for valid system call.
- Invokes specific service routine associated with syscall number (contained in `eax`):
 - `call *sys_call_table(0, %eax, 4)`
- Return code of system call is stored in `eax`.

Parameter Passing

- On the 32-bit Intel 80x86:
 - 6 registers are used to store syscall parameters.
 - **eax** (syscall number).
 - **ebx, ecx, edx, esi, edi** store parameters to syscall service routine, identified by syscall number.

Wrapper Routines

- Kernel code (e.g., kernel threads) cannot use library routines.
- `_syscall10 ... _syscall15` macros define wrapper routines for system calls with up to 5 parameters.
- e.g., `_syscall13(int,write,int,fd,
 const char *,buf,unsigned int,count)`

Example: “Hello, world!”

```
.data                                # section declaration

msg:
    .string "Hello, world!\n"        # our dear string
    len = . - msg                    # length of our dear string

.text                                  # section declaration

    # we must export the entry point to the ELF linker or
.global _start                        # loader. They conventionally recognize _start as their
    # entry point. Use ld -e foo to override the default.

_start:

# write our string to stdout

    movl    $len,%edx                # third argument: message length
    movl    $msg,%ecx                # second argument: pointer to message to write
    movl    $1,%ebx                  # first argument: file handle (stdout)
    movl    $4,%eax                  # system call number (sys_write)
    int     $0x80                    # call kernel

# and exit

    movl    $0,%ebx                  # first argument: exit code
    movl    $1,%eax                  # system call number (sys_exit)
    int     $0x80                    # call kernel
```

Linux Files Relating to Syscalls

- Main files:
 - arch/i386/kernel/entry.S
 - System call and low-level fault handling routines.
 - include/asm-i386/unistd.h
 - System call numbers and macros.
 - kernel/sys.c
 - System call service routines.

arch/i386/kernel/entry.S

```
.data
ENTRY(sys_call_table)
.long SYMBOL_NAME(sys_ni_syscall) /* 0 - old "setup()" system
                                   call*/
    .long SYMBOL_NAME(sys_exit)
    .long SYMBOL_NAME(sys_fork)
    .long SYMBOL_NAME(sys_read)
    .long SYMBOL_NAME(sys_write)
```

- Add system calls by appending entry to `sys_call_table`:

```
.long SYMBOL_NAME(sys_my_system_call)
```

include/asm-i386/unistd.h

- Each system call needs a number in the system call table:
 - e.g., `#define __NR_write 4`
 - `#define __NR_my_system_call nnn`, where `nnn` is next free entry in system call table.

kernel/sys.c

- Service routine bodies are defined here:
- e.g., `asmlinkage retval`

```
sys_my_system_call (parameters) {  
    body of service routine;  
    return retval;  
}
```

Kernel Modules



Kernel Modules

- See A. Rubini, “Device Drivers”, Chapter 2.
- Modules can be compiled and dynamically linked into kernel address space.
 - Useful for device drivers that need not always be resident until needed.
 - Keeps core kernel “footprint” small.
 - Can be used to “extend” functionality of kernel too!

Example: “Hello, world!”

```
#define MODULE
#include <linux/module.h>
int init_module(void) {
    printk("<1>Hello, world!\n");
    return 0;
}
void cleanup_module(void) {
    printk("<1>Goodbye cruel world 😞\n");
}
```


Using Modules

- Module object file is installed in running kernel using `insmod module_name`.
 - Loads module into kernel address space and links unresolved symbols in module to symbol table of running kernel.

The Kernel Symbol Table

- Symbols accessible to kernel-loadable modules appear in `/proc/ksyms`.
 - `register_syntab` registers a symbol table in the kernel's main table.
 - Real hackers export symbols from the kernel by modifying `kernel/ksyms.c` 😊

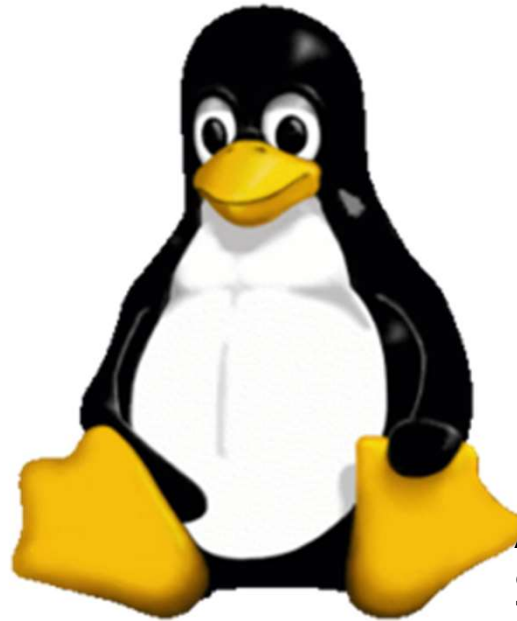
Project Suggestions (1)

- Real-Time thread library.
- Scheduler activations in Linux.
- A Linux “upcall” mechanism.
- Real-Time memory allocator / garbage collector.
- A distributed shared memory system.
- A QoS-based socket library.
- An event-based mechanism for implementing adaptive systems.
- DWCS packet scheduling.
- A heap-based priority scheduler for Linux.

Project Suggestions (2)

- μ S resolution timers for Linux.
- Porting the Bandwidth-Broker to Linux.
- A QoS Management framework like QuO or Dionisys.
- A Real-Time communications protocol.
- A feedback-control system for flow/error/rate/congestion control.
- “Active Messages” for Linux.
- A thread continuation mechanism.
- A thread migration / load-balancing system.

Introduction to Linux

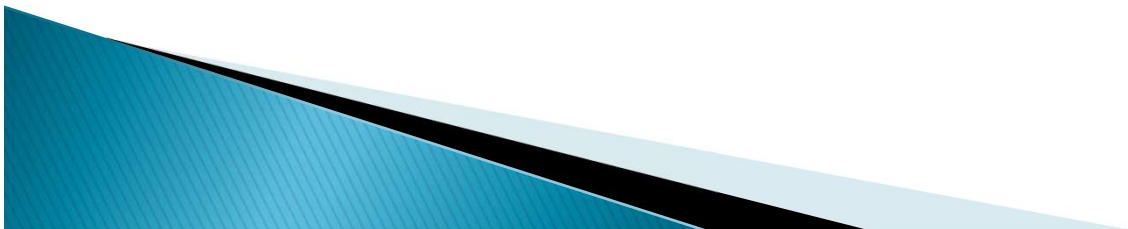


Dr. M. Durairaj
Associate Professor
School of Computer Science,
Engineering and Applications
Bharathidasan University

“Linux at the Command Line”
Don Johnson of BU IS&T

About the class...

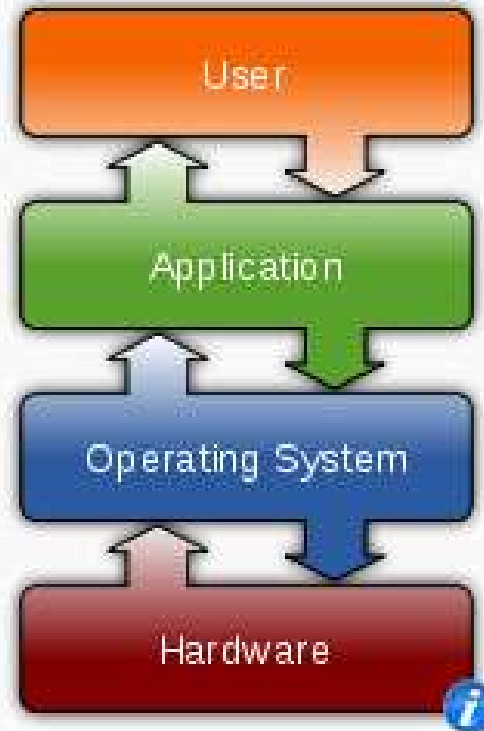
- ▶ We'll start with a sign in sheet that include questions about your Linux experience and goals.
- ▶ We'll end with a class evaluation.
- ▶ We'll cover as much as we can in the time allowed, starting with the easiest and most important material. Don't feel rushed; if we don't cover everything, you'll pick it up as you continue working with Linux.
- ▶ This is a hands-on, lab class; ask questions at any time.
- ▶ Commands for you to type are in **BOLD**
- ▶ We'll take a break at the half-way point.



What is Linux?

It's an Operating System

Operating systems



Common features

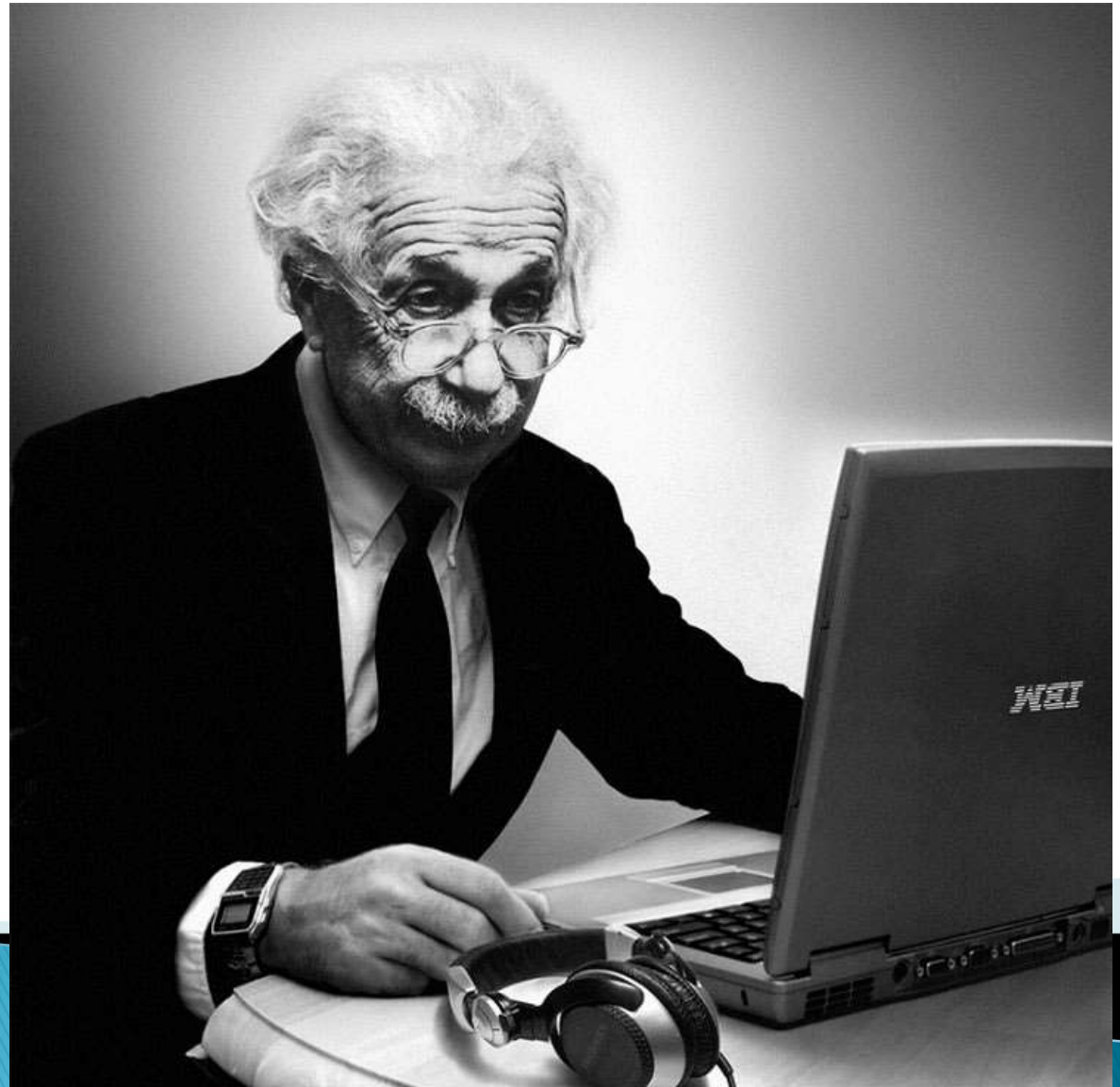
- Process management
- Interrupts
- Memory management
- File system
- Device drivers
- Networking (TCP/IP, UDP)
- Security (Process/Memory protection)
- I/O



What is Linux?



The Most
Common O/S
Used By BU
Researchers When
Working on a
Server or
Computer Cluster




What is Linux?

- ▶ Linux is a Unix clone written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net.
- ▶ Unix is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs.
- ▶ Linux and Unix strive to be POSIX compliant.
- ▶ 64% of the world's servers run some variant of Unix or Linux. The Android phone and the Kindle run Linux.



The Linux Philosophy

*The *Nix Philosophy of Doug McIlroy*

- (i) Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.
 - (ii) Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
 - (iii) Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.
- 

Linux Has Many Distributions



Linux Has Many Distributions

BU uses CentOS in its Linux cluster which is a free version of RedHat Enterprise Linux with the trademarks removed



What is Linux?

Linux + GNU Utilities = Free Unix



- ▶ Linux is an O/S core written by Linus Torvalds and others AND




- ▶ a set of small programs written by Richard Stallman and others. They are the GNU utilities.

<http://www.gnu.org/>

What is Linux?

“Small programs that do one thing well”
(see [unix-reference.pdf](#))

- ▶ Network: ssh, scp, ping, telnet, nslookup, wget
 - ▶ Shells: BASH, TCSH, alias, watch, clear, history, chsh, echo, set, setenv, xargs
 - ▶ System Information: w, whoami, man, info, which, free, echo, date, cal, df, free, man, info
 - ▶ Command Information: man, info
 - ▶ Symbols: |, >, >>, <, &, >&, 2>&1, ;, ~, ., .., \$!, !:<n>, !<n>
 - ▶ Filters: grep, egrep, more, less, head, tail
 - ▶ Hotkeys: <ctrl><c>, <ctrl><d>
 - ▶ File System: ls, mkdir, cd, pwd, mv, ln, touch, cat, file, find, diff, cmp, /net/<hostname>/<path>, mount, du, df, chmod, find
 - ▶ Line Editors: awk, sed
 - ▶ File Editors: vim, gvim, emacs -nw, emacs
- 

What is Linux?

“Small programs that do one thing well”

We will not cover the commands below in this class, but you need to know them. See the man pages for the process commands and the “sge” folder inside of the “cheat sheets and tutorials” folder for the SGE (Sun Grid Engine) command tutorials: `qsh-interactive.pdf`, `qsh-interactive-matlab.pdf`, `qsub-batch.pdf`, `qsub-batch-matlab.pdf`, and `qstat-qhost.pdf`.

- ▶ Process Management: `ps`, `top`, `kill`, `killall`, `fg`, `bg`
- ▶ SGE Cluster: `qsh`, `qstat`, `qsub`, `qhost`

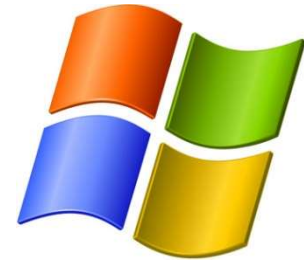


Connecting to a Linux Host

- ▶ You need a “xterm” emulator: software that emulates an “X” terminal and connects using the “SSH” secure shell protocol.
- ▶ You are sitting at the “client,” either a Windows, Macintosh or even possibly a Linux machine.
- ▶ You are connecting to a “server,” typically the “head” or “gateway” node of a cluster of computers. You will be working on the head node or submitting jobs to execution nodes, all of them, Linux machines.
- ▶ You can also connect to a Linux machine by using VNC to get a whole desktop if it’s supported by the server.

Connecting to a Linux Host – Windows Client Software

- ▶ You need a “xterm” emulation – software that emulates an “X” terminal and that connects using the “SSH” Secure Shell protocol.
 - Windows
 - If you don’t need windowing, “putty” is good:
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
 - If you need windowing, use StarNet “X-Win32:”
<http://www.bu.edu/tech/desktop/site-licensed-software/xwindows/xwin32/>



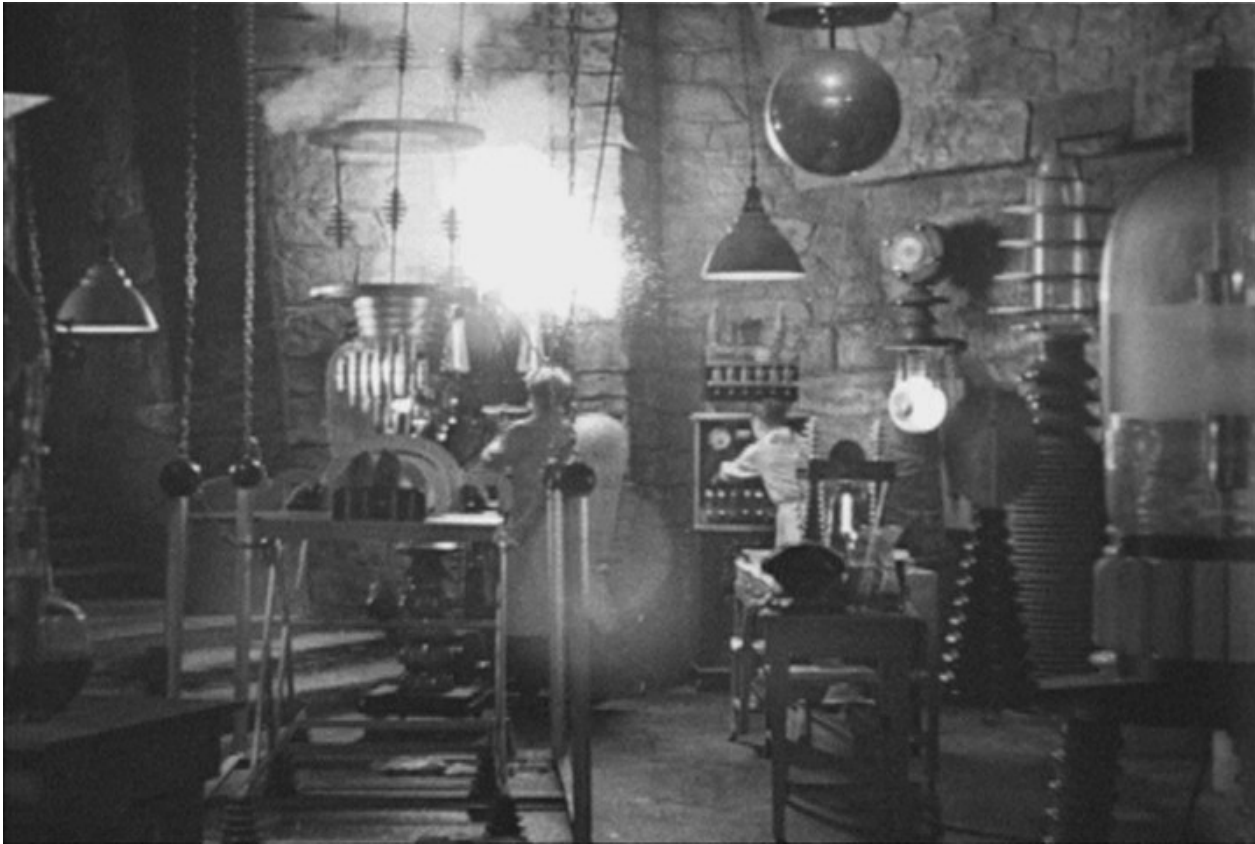
Connecting to a Linux Host – Mac OS X Client Software



- Mac OS X
 - “Terminal” is already installed
 - Why? Darwin, the system on which Apple's Mac OS X is built, is a derivative of 4.4BSD-Lite2 and FreeBSD. In other words, the Mac is a Unix system!



Let the Linux Lab Begin!

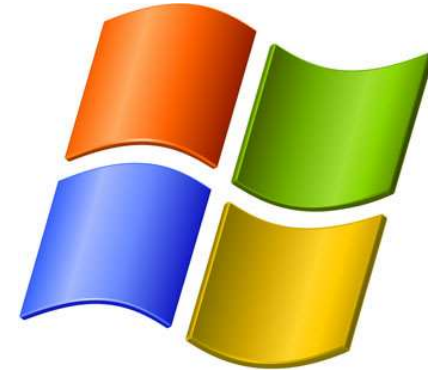


The Ideal Lab Facility

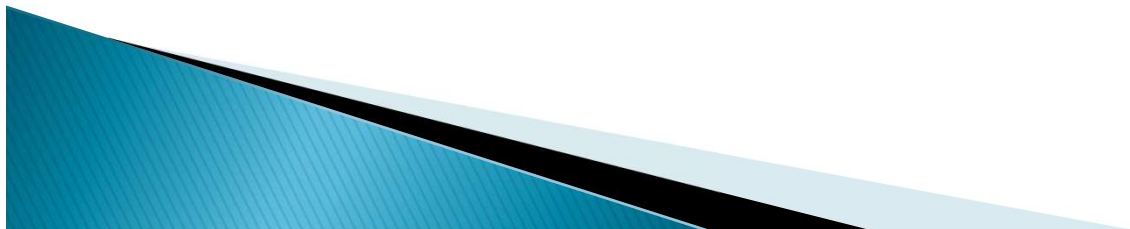


Your Instructor Today

Connecting to a Linux Host – Windows Client

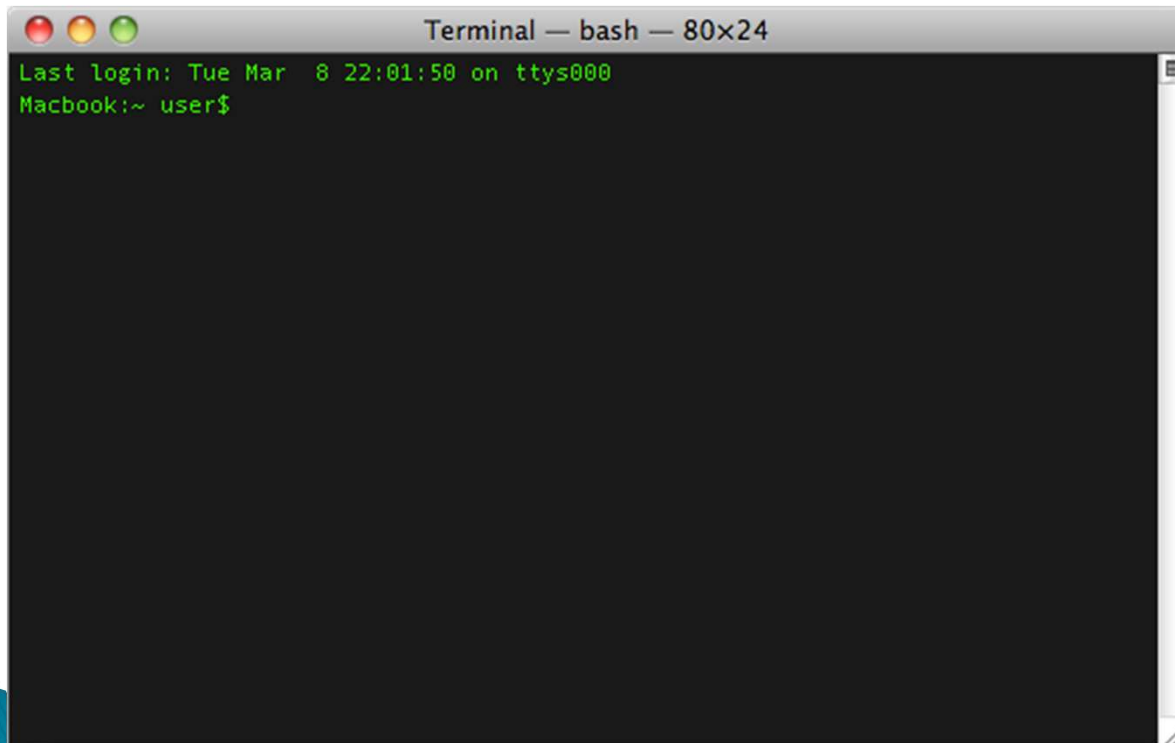


- ▶ X-Win32 / X-Config
 - Wizard
 - Name: katana
 - Type: ssh
 - Host: katana.bu.edu (Off-campus, must include domain “bu.edu”)
 - Login: <userID>
 - Password: <password>
 - Command: Linux
 - Click “katana” then “Launch”
 - Accept the host server public key (first time only)



Connecting to a Linux Host – Mac OS X Client

- ▶ Terminal
 - Type `ssh -X katana.bu.edu` or `ssh -Y katana.bu.edu` (less secure)

A screenshot of a Mac OS X Terminal window. The title bar reads "Terminal — bash — 80x24". The terminal content shows a successful login session: "Last login: Tue Mar 8 22:01:50 on ttys000" followed by the prompt "Macbook:~ user\$".

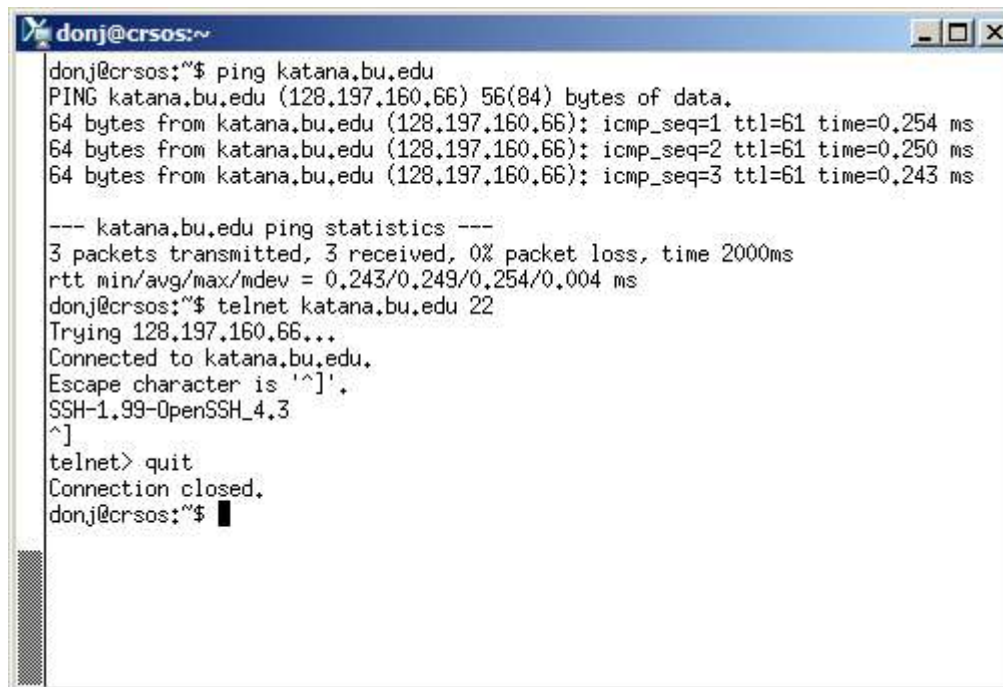
```
Terminal — bash — 80x24
Last login: Tue Mar 8 22:01:50 on ttys000
Macbook:~ user$
```



Connection Problems

When there are problems connecting to a login host, try:

- ▶ ping katana.bu.edu
- ▶ telnet katana.bu.edu 22



```
donj@crsos:~  
donj@crsos:~$ ping katana.bu.edu  
PING katana.bu.edu (128.197.160.66) 56(84) bytes of data.  
64 bytes from katana.bu.edu (128.197.160.66): icmp_seq=1 ttl=61 time=0.254 ms  
64 bytes from katana.bu.edu (128.197.160.66): icmp_seq=2 ttl=61 time=0.250 ms  
64 bytes from katana.bu.edu (128.197.160.66): icmp_seq=3 ttl=61 time=0.243 ms  
  
--- katana.bu.edu ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 0.243/0.249/0.254/0.004 ms  
donj@crsos:~$ telnet katana.bu.edu 22  
Trying 128.197.160.66...  
Connected to katana.bu.edu.  
Escape character is '^]'.  
SSH-1.99-OpenSSH_4.3  
^]  
telnet> quit  
Connection closed.  
donj@crsos:~$
```

Obtaining the Course Material

- ▶ Windows

- Using File Explorer, copy the directory “\\scv-files.bu.edu\SCV\Training\Introduction to Linux” to “My Documents” on your lab machine

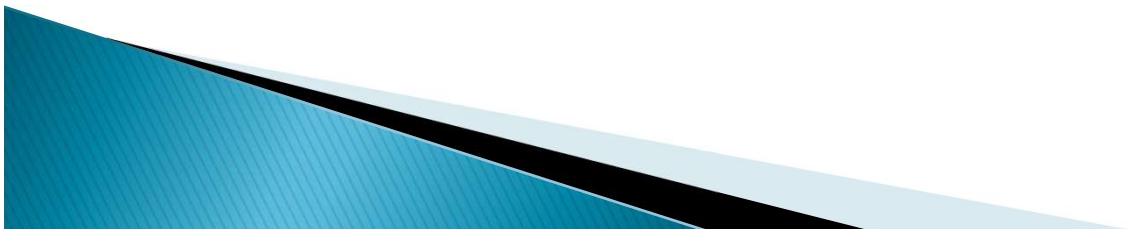
- ▶ Linux

- Connect to `katana.bu.edu` using X-Win32 and run this command:
 - `cp -Rv /project/ssrcsupp/linux_class ~/`



Connecting to a Linux Host: Emulate a Browser

- ▶ Note: <CR> is short for “carriage return” and equals the ASCII press the “Enter” or “Return” key. It tells the shell that you finished sending one line (see [ascii-table.pdf](#)).
- ▶ Try
 - telnet www.bu.edu
 - GET / HTTP/1.1
 - Host:www.bu.edu<CR>
 - <CR>
- ▶ What happened?




```
xterm
tuta0@katana:~$ telnet www.bu.edu 80
Trying 128.197.26.34...
Connected to www.bu.edu.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.bu.edu
█
```

Connecting to a Linux Host >>

Emulate a Browser

Connecting to a Linux Host: Send and Email

- ▶ Try
 - telnet localhost 25
 - ehlo me
 - mail from:<your email address>
 - rcpt to:<destination email address>
 - data
 - Subject:<subject of email>
 - <Body of email>
 - .
 - <CR>
- ▶ What Happened?




```
xterm
tuta0@katana:~$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 katana.bu.edu ESMTF Postfix
ehlo me
250-katana.bu.edu
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from:donj@bu.edu
250 2.1.0 Ok
rcpt to:amydonj@gmail.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: Hello
This is sending email the hard way!
*
250 2.0.0 Ok: queued as 1CDAA48063
```

Connecting to an Linux Host >>

Send and Email

The Shell

- ▶ A shell is a computer program that interprets the commands you type and sends them to the operating system. Secondly, it provide a programming environment consisting of environment variables.
 - ▶ Most BU systems, including the BU Linux Cluster, support at least two shells: TCSH and BASH. The default shell for your account is TCSH. The most popular and powerful Linux shell today is BASH.
 - ▶ To determine your shell type:
 - `echo $SHELL` (shell prints contents of env)
 - `echo "$SHELL"` (shell still processes env. variable)
 - `echo '$SHELL'` (shell treats env. variable as simple literal)
 - ▶ The complete environment can be printed with `set`, `setenv` (TCSH) and `set` (BASH).
 - ▶ To determine the path to the shell program, type:
 - `which bash`
 - `which tcsh`
 - ▶ Change the shell with `chsh /bin/bash` (provide path to new shell as a "parameter," meaning to be explained soon)
- 

```
xterm
tuta0@katana:~$ echo $SHELL
/bin/bash
tuta0@katana:~$ which tcsh
/bin/tcsh
tuta0@katana:~$ which bash
/bin/bash
tuta0@katana:~$ chsh /bin/bash
Old password:
Your current shell already is /bin/bash.
tuta0@katana:~$ chsh /bin/tcsh
Old password:
Changing shell for tuta0...
succeeded. The change will take effect within 15 minutes on all SCF systems.
tuta0@katana:~$
```

The Shell >>>

Output of the echo, which and chsh commands

System Information

- ▶ After you connect, type
 - shazam
 - whoami
 - hostname
 - date
 - cal
 - free
- ▶ Commands have three parts; *command*, *options* and *parameters*. Example: `cal -j 3 1999`. “cal” is the command, “-j” is an option (or switch), “3” and “1999” are parameters.
- ▶ Options have long and short forms. Example:
 - `date -u`
 - `date --universal`

What is the nature of the prompt?

What was the system’s response to the command?



```
xterm
tuta0@katana:~$ shazam
-bash: shazam: command not found
tuta0@katana:~$ whoami
tuta0
tuta0@katana:~$ hostname
katana
tuta0@katana:~$ date
Sun Sep  9 12:14:51 EDT 2012
tuta0@katana:~$ cal
  September 2012
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
tuta0@katana:~$ free
              total        used         free       shared    buffers     cached
Mem:           8109704       7759508         350196            0        100204       5794856
-/+ buffers/cache:    1864448         6245256
Swap:          9244240         502656         8741584
tuta0@katana:~$ date -u
Sun Sep  9 16:15:01 UTC 2012
tuta0@katana:~$ date --universal
Sun Sep  9 16:15:07 UTC 2012
tuta0@katana:~$ █
```

System Information >>

Output of the whoami, hostname, date, cal and free

Command History and Simple Command Line Editing

- ▶ Try the `history` command
- ▶ Try `<Ctrl><r>` (only works in BASH shell)
- ▶ Choose from the command history by using the up `↑` and down `↓` arrows
- ▶ What do the left `←` and right `→` arrow do on the command line?
- ▶ Try the `` and `<Backspace>` keys



Help with Commands

- ▶ Type
 - `hostname --help`
 - `man hostname`
 - `info hostname` (gives the same or most information, but must be paged)
- ▶ And “Yes,” you can always Google it



Connect Commands Together with the Pipe Symbol “|” and Using Filters

- ▶ The pipe “|” feeds the OUTPUT of one command into the INPUT of another command. Our first example will use the pipe symbol to filter the output of a command. Try:
 - `w`
 - `w | grep 'root'`
 - `ps -e -o ruser,comm | grep 'tut'`
- ▶ The `ps` command is using both “options (dash)” and parameters
- ▶ Try both “`man grep`” and “`info grep`”. See the difference?



Editing the Command Line with Emacs Keys (see [emacs-editing-mode.pdf](#))

- ▶ `<Ctrl-a>` go to beginning
- ▶ `<Ctrl-e>` go to end
- ▶ `<Alt-f>` forward one word
- ▶ `<Alt-b>` back one word
- ▶ `<Ctrl-f>` forward one character
- ▶ `<Ctrl-b>` back one character
- ▶ `<Ctrl-d>` delete character
- ▶ `<Alt-d>` delete word
- ▶ `<Ctrl-u>` delete from cursor to beginning of line
- ▶ `<Ctrl-k>` delete from cursor to end of line

See [emacs-editing-mode.pdf](#) and [emacs-editing-mode-short.pdf](#)

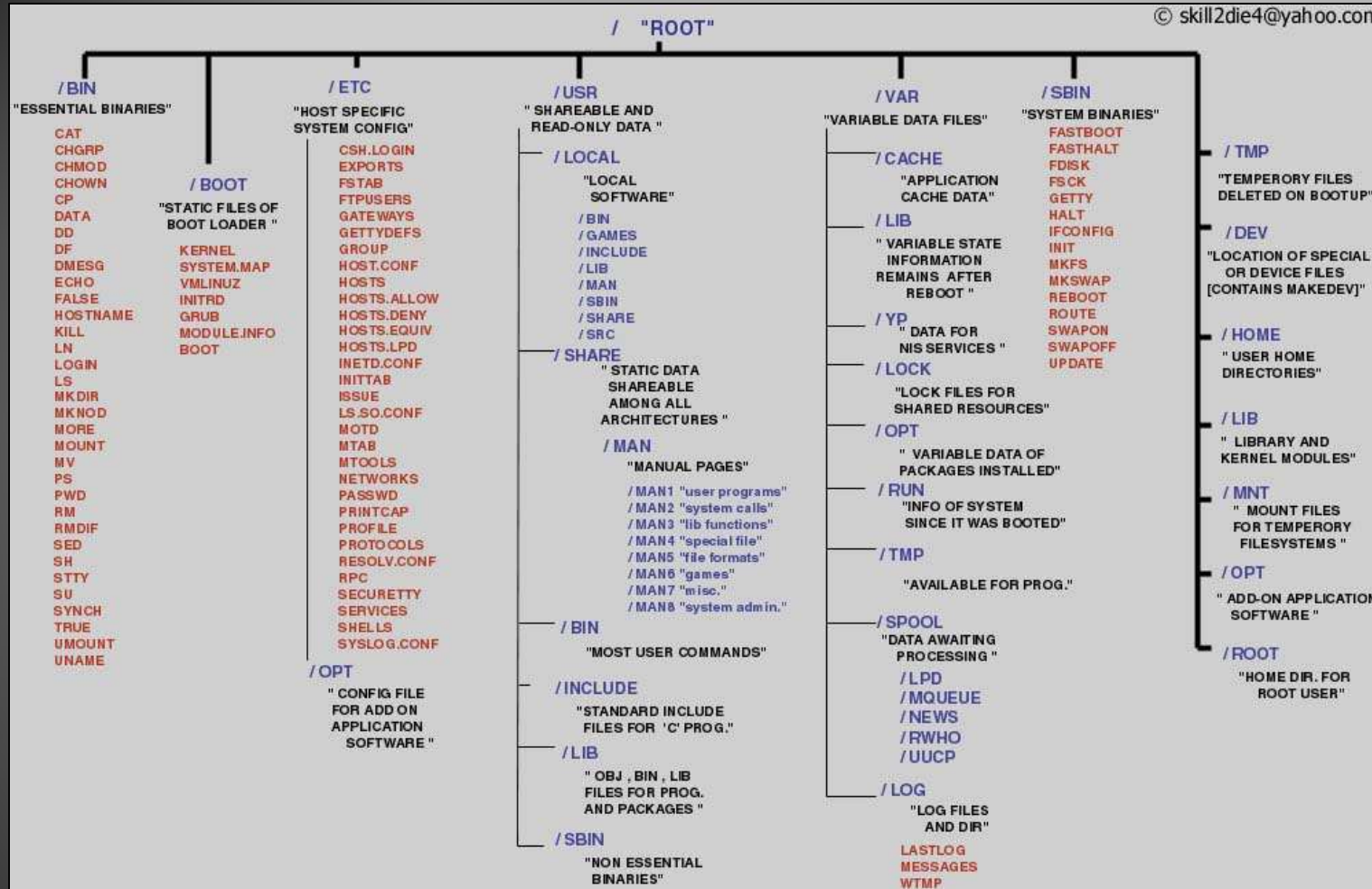
Go to through command history in shell and practice editing.



The Linux File System

- ▶ The *Nix (Unix or Linux) file system is a hierarchical directory structure
- ▶ The structure resembles an upside down tree
- ▶ Directories are collections of files and other directories. The structure is recursive with many levels.
- ▶ Every directory has a parent except for the root directory.
- ▶ Many directories have children directories.
- ▶ Unlike Windows, with multiple drives and multiple file systems, a *Nix system only has ONE file system.
- ▶ The Linux Standard Base (LSB) specifies the structure of a Linux file system.





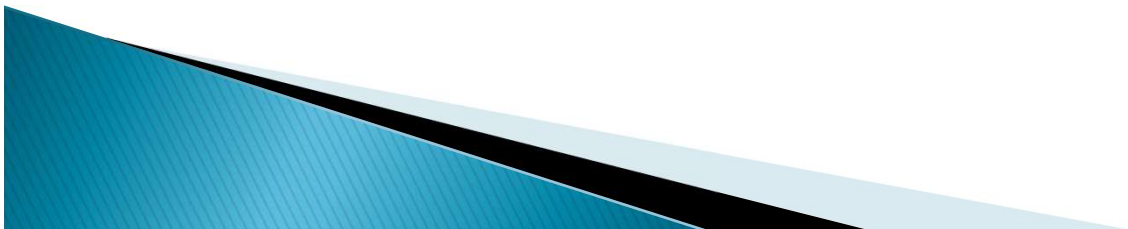
The Linux File System >>

A Typical Linux File System

Examining the File System

▶ Try

- `nautilus --browser --no-desktop`
- `tree -L 3 -d / | less`
- `tree -L 3 / | less`
- `file /bin/alsac` then press `<tab>`
- `cd ~; pwd` (This is your home directory where application settings are kept and where you have write privileges)
- `ls`
- `mkdir myPics;mkdir myPics/work;mkdir myPics/friends;mkdir myPics/friends/BU; mkdir myPics/friends/MIT`
- `tree myPics`



```
xterm
katana:~ % tree -L 3 -d / | head -3
/
|-- bin
|-- boot
katana:~ % tree -L 3 / | head -3
/
|-- bin
|   |-- alsacard
katana:~ % file /bin/alsacard
/bin/alsacard: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/
Linux 2.6.9, dynamically linked (uses shared libs), for GNU/Linux 2.6.9, stripp
d
katana:~ % cd ~ ; pwd
/usr4/tutorial/tuta0
katana:~ % ls
Desktop/          bupage.html      helloworld.c     untitled folder/
bin/              helloworld*      test.sh*
katana:~ % mkdir myPics ; mkdir myPics/work ; mkdir myPics/friends ; mkdir myPic
s/friends/BU ; mkdir myPics/friends/MIT
katana:~ % tree myPics
myPics
|-- friends
|   |-- BU
|   |-- MIT
|-- work

4 directories, 0 files
katana:~ % █
```

Examining the File System >>

Output from the tree, file, pwd and ls commands
Demonstration of using the mkdir command

Navigating the File System

- ▶ There are two types of pathnames
 - Absolute (Abs) – the full path to a directory or file; begins with the root symbol /
 - Relative (Rel) – a partial path that is relative to the current working directory
- ▶ Examples
 - Abs `cd /usr/local/lib`
 - `echo $HOME` (one of many environment variables maintained by the shell)
 - Abs `cd `echo $HOME``
 - `pwd`
 - Rel `cd ..`
 - Rel `cd ..`
 - Abs `cd /lib` (location OS shared libraries)
 - `ls -d */` (a listing of only the directories in /lib)




```
xterm
tuta0@katana:/lib> cd /usr/local/lib
tuta0@katana:/usr/local/lib> echo $HOME
/usr4/tutorial/tuta0
tuta0@katana:/usr/local/lib> cd `echo $HOME`
tuta0@katana:~> pwd
/usr4/tutorial/tuta0
tuta0@katana:~> cd ..
tuta0@katana:/usr4/tutorial> cd ..
tuta0@katana:/usr4> cd /lib
tuta0@katana:/lib> ls -d */
bdev/    firmware/ kbd/    modules/ security/
dbus-1.0/ i686/    lsb/    rtkai/   udev/
tuta0@katana:/lib> █
```

Navigating the File System >>

Moving around the file system using the cd command

Modifying the Linux File System

- ▶ More useful commands
 - `cd` (also takes you to your home directory like `cd ~`)
 - `mkdir test`
 - `echo 'Hello everyone' > test/myfile.txt`
 - `echo 'Goodbye all' >> test/myfile.txt`
 - `less test/myfile.txt`
 - `mkdir test/subdir1 /subdir2` (FAILS)
 - `mkdir -p test/subdir1 /subdir2` (Succeeds)
 - `mv test/myfile.txt test/subdir1 /subdir2`
 - `rmdir test` (FAILS)
 - `rm -Rv test` (Succeeds)



```
xterm
tuta0@katana:~$ cd
tuta0@katana:~$ mkdir test
tuta0@katana:~$ echo 'Hello everyone' > test/myfile.txt
tuta0@katana:~$ echo 'Goodbye everyone' >> test/myfile.txt
tuta0@katana:~$ less test/myfile.txt
tuta0@katana:~$ mkdir test/subdir1/subdir2
mkdir: cannot create directory `test/subdir1/subdir2': No such file or directory
tuta0@katana:~$ mkdir -p test/subdir1/subdir2
tuta0@katana:~$ mv test/myfile.txt test/subdir1/subdir2/
tuta0@katana:~$ rmdir test
rmdir: test: Directory not empty
tuta0@katana:~$ rm -Rv test
removed `test/subdir1/subdir2/myfile.txt'
removed directory: `test/subdir1/subdir2'
removed directory: `test/subdir1'
removed directory: `test'
tuta0@katana:~$
```

Modifying the Linux File System >>

Demonstration of the mkdir, less, mv, rmdir and rm commands

The List Command

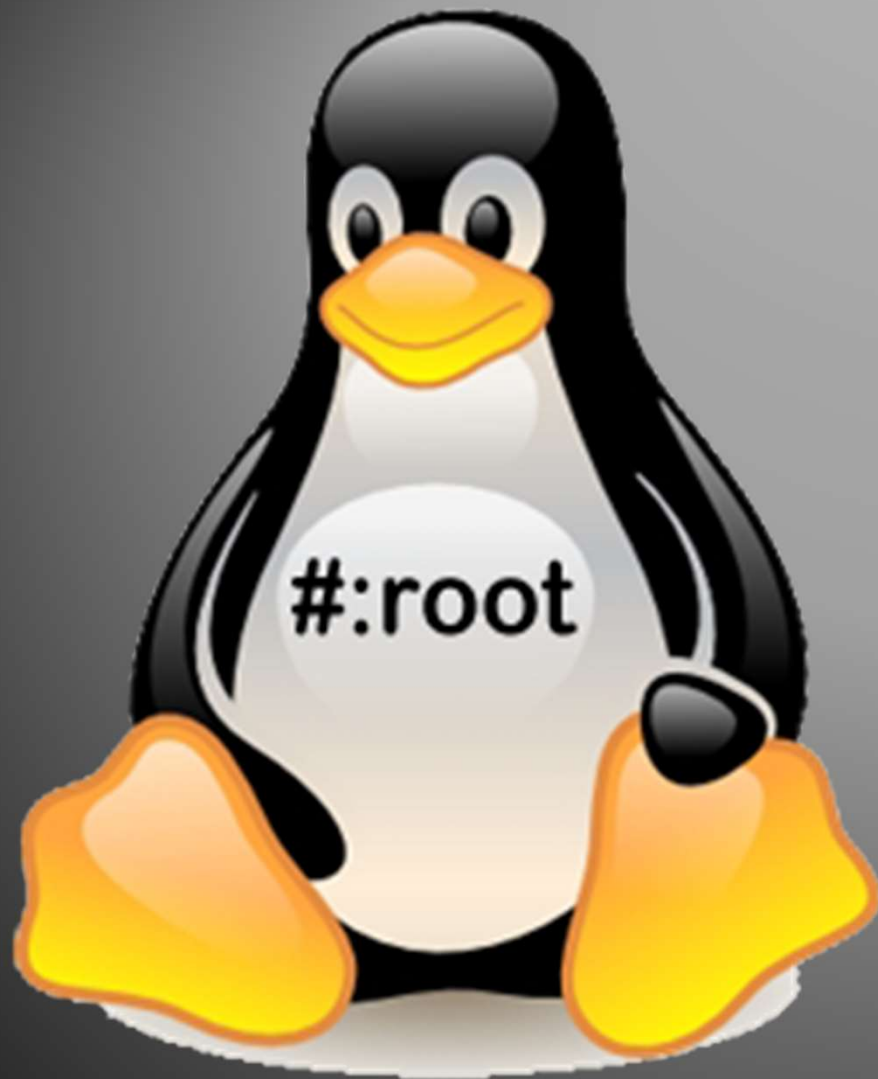
- ▶ Useful options for the “ls” command:
 - `ls -a` List all file including hidden file beginning with a period “.”
 - `ls -ld *` List details about a directory and not its contents
 - `ls -F` Put an indicator character at the end of each name
 - `ls -l` Simple long listing
 - `ls -lh` Give human readable file sizes
 - `ls -lS` Sort files by file size
 - `ls -lt` Sort files by modification time



File System Ownership and Permissions

- ▶ All files and directories have a individual and a group *ownership*.
- ▶ All files and directories have read (r), write (w), and execute (x) *permissions* assigned as octets to the individual owner (u), the group (g) owner and all others (o) that are logged into the system.
- ▶ You can change permissions if you are the individual owner or a member of the group.
- ▶ Only root can change ownership.





root >>

The root user is the master

File and Directory Ownership and Permissions

- ▶ Try
 - `cd`
 - `touch myfile` (create file)
 - `mkdir mydir` (create directory)
 - `ls -l myfile` (examine file)
 - `ls -ld mydir` (examine directory)
 - `chmod g+w myfile` (add group write permission)
 - `ls -l myfile`
 - `chmod ugo+x myfile` (add user, group and other execute permission)
 - `ls -l myfile`
 - `chmod ugo+w mydir` (add user, group and other write permission)
 - `ls -ld mydir`
 - `chmod a-w` (a=ALL, remove user, group and other write permission)



```
xterm
tuta0@katana:~$ cd /usr/local
tuta0@katana:/usr/local$ cd
tuta0@katana:~$ touch myfile
tuta0@katana:~$ mkdir mydir
tuta0@katana:~$ ls -l myfile
-rw-r--r-- 1 tuta0 tutorial 0 Sep  9 13:39 myfile
tuta0@katana:~$ ls -ld mydir
drwxr-xr-x 2 tuta0 tutorial 4096 Sep  9 13:39 mydir
tuta0@katana:~$ chmod g+w myfile
tuta0@katana:~$ ls -l myfile
-rw-rw-r-- 1 tuta0 tutorial 0 Sep  9 13:39 myfile
tuta0@katana:~$ chmod ugo+x myfile
tuta0@katana:~$ ls -l myfile
-rwxrwxr-x 1 tuta0 tutorial 0 Sep  9 13:39 myfile
tuta0@katana:~$ chmod ugo+w mydir
tuta0@katana:~$ ls -ld mydir
drwxrwxrwx 2 tuta0 tutorial 4096 Sep  9 13:39 mydir
tuta0@katana:~$ chmod a-w mydir
tuta0@katana:~$ ls -ld mydir
dr-xr-xr-x 2 tuta0 tutorial 4096 Sep  9 13:39 mydir
tuta0@katana:~$
```

File and Directory Ownership and Permissions >>

Examining and changing file and directory permissions

Editing Output Lines With *awk* (see [awk-tutorial.pdf](#))

- ▶ Syntax:

```
BEGIN { Actions }
```

```
{ACTION} # Action for every line in a file
```

```
END { Actions }
```

- ▶ Try

- `ls -l /usr`

- `ls -l /usr | awk '{print $9 "\t" $5}'`

- `ls -l /usr > usr.txt`

- `awk '{print $9 "\t" $5}' usr.txt` (gives same results as 2nd command line, but `awk` is acting on a file instead of saved output)

- `ls -lh /lib | awk '{printf "%20s\t%s\n", $9, $5}'`

- `ls -l /lib | awk 'BEGIN {sum=0} {printf "%20s\t%s\n", $9, $5; sum+= $5} END{sum/=1000000; printf "\nTotal: %d GB\n", sum}'`



```
donj@crsos:~  
tuta0@katana:~> ls -l /usr | tail -3  
drwxr-xr-x 245 root root 12288 Aug  4 17:09 share  
drwxr-xr-x  5 root root  4096 Aug  4 17:08 src  
lrwxrwxrwx  1 root root   10 Dec  3 2007 tmp -> ../var/tmp  
tuta0@katana:~> ls -l /usr | awk '{print $9 "\t" $5}' | tail -3  
share 12288  
src 4096  
tmp 10  
tuta0@katana:~> ls -l /lib | awk '{printf "%20s\t%s\n", $9, $5}' | tail -3  
          rtkaio 4096  
          security 4096  
          udev 4096  
tuta0@katana:~> ls -l /lib | awk 'BEGIN {sum=0} {printf "%20s\t%s\n", $9, $5; sum+=  
=$5} END{sum/=1000000; printf "\nTotal: %d GB\n", sum}' | tail -7  
          lsb 4096  
          modules 4096  
          rtkaio 4096  
          security 4096  
          udev 4096  
  
Total: 17 GB  
tuta0@katana:~> █
```

Editing Output Lines With awk >>

Output from awk commands

Editing Output Lines With *sed*

- ▶ *sed* replaces one substring with another
- ▶ *sed* operates on every line in a file or processes every line piped into it
- ▶ *sed* matches patterns using *regular expressions* (See *regular-expressions.pdf* cheat sheet)
- ▶ Common regular expression metacharacters:
 - . – any character
 - ? – quantified zero or one
 - * – quantifier none or more
 - + – quantifier one or more
 - ^ – beginning of line
 - \$ – end of line
 - [XxYy] – character class matching upper or lower case “X” or “Y”



Editing Output Lines With *sed* – continued (see [sed-tutorial.pdf](#))

▶ Try

- `echo "The rain in Spain stays mainly in the plain." > easy_sed.txt; cat easy_sed.txt`
- `sed -i.bak 's/rain/snow/;s/Spain/Sweden/;s/plain/mountains/' easy_sed.txt; cat easy_sed.txt`
- `ls -l /lib | awk 'BEGIN {sum=0} {printf "%s\t%s\n", $9, $5; sum += $5} END {printf "\nTotal: %d\n", sum}' | sed -e 's/\.so\(\.[0-9]*\)*//' | less`
(challenge: get rid of soname extension)
- `ls -l /lib | awk 'BEGIN {sum=0} {printf "%s\t%s\n", $9, $5; sum += $5} END {printf "\nTotal: %d GB\n", sum}' | sed -e 's/\.so\(\.[0-9]*\)*//' | awk '{printf "%20s\t%s\n", $1, $2}' | less` (pretty print)

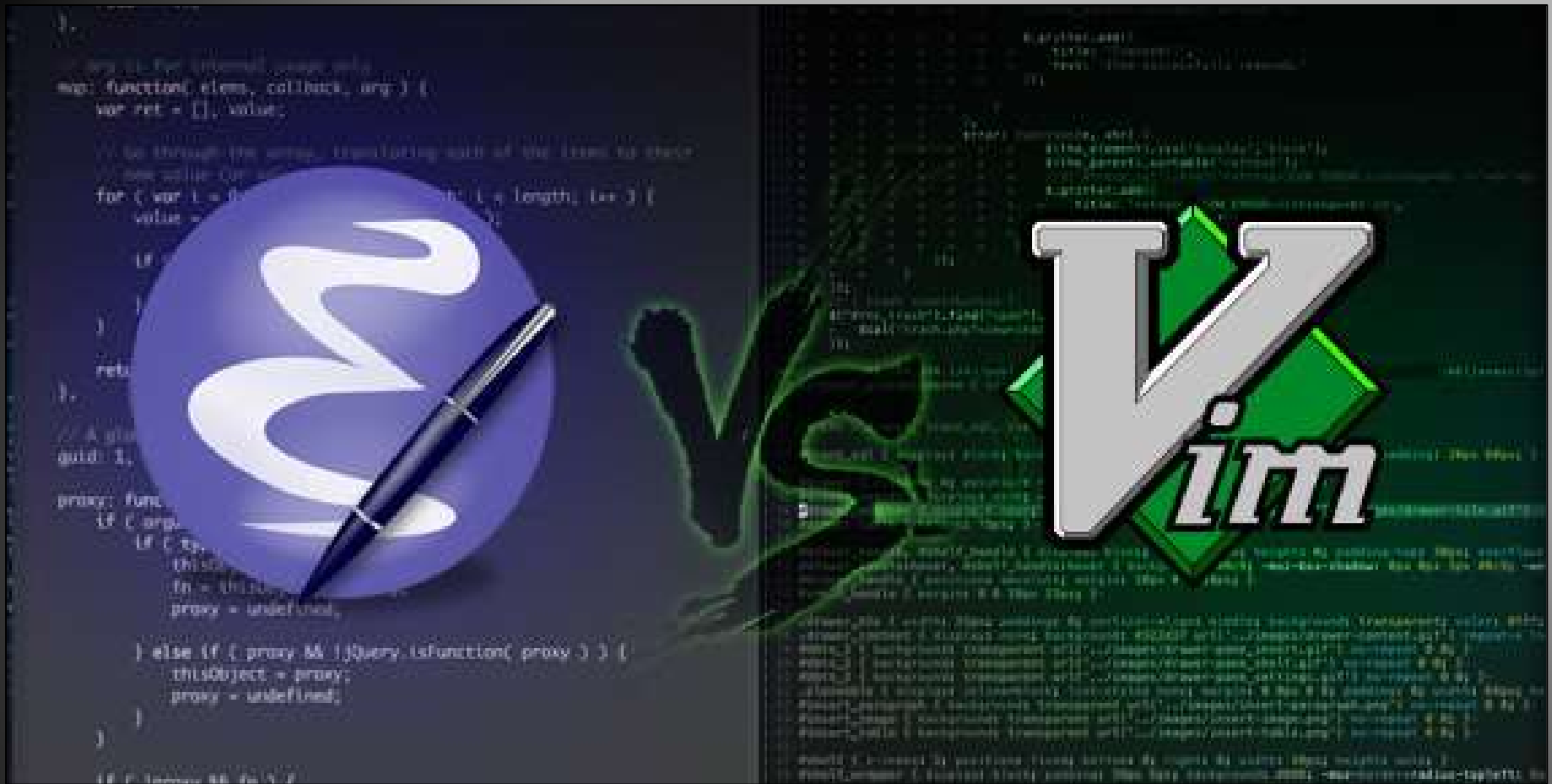
```
donj@crsos:~
tuta0@katana:~$ echo "The rain in Spain stays mainly in the plain." > easy_sed.txt
tuta0@katana:~$ cat easy_sed.txt
The rain in Spain stays mainly in the plain.
tuta0@katana:~$ sed -i.bak 's/rain/snow;/s/Spain/Sweden;/s/plain/mountains/' easy_sed.txt
tuta0@katana:~$ cat easy_sed.txt
The snow in Sweden stays mainly in the mountains.
tuta0@katana:~$ ls -l /lib | awk 'BEGIN {sum=0} {printf "%s\t%s\n", $9, $5; sum+=$5} END{printf "\nTotal: %d\n", sum}' | sed -e 's/\.,so\(\.[0-9]*\)*/' | tail -7
lsb      4096
modules 4096
rtkaio   4096
security 4096
udev     4096

Total: 17279594
tuta0@katana:~$ ls -l /lib | awk 'BEGIN {sum=0} {printf "%s\t%s\n", $9, $5; sum+=$5} END{printf "\nTotal: %d GB\n", sum}' | sed -e 's/\.,so\(\.[0-9]*\)*/' | awk '{printf "%20s\t%s\n", $1, $2}' | tail -7
lsb      4096
modules  4096
rtkaio    4096
security  4096
udev      4096

Total:    17279594
tuta0@katana:~$
```

Editing Output Lines With sed >>

Output from sed commands



Editing Files with Emacs and Vim >>

You don't have to take sides and there is always "nedit"

Editing Files with Emacs and Vim

- ▶ Cheat sheet: [emacs.pdf](#)
- ▶ Movement: `<C-b>`, `<C-n>`, `<C-p>`, `<C-f>`, `<M-b>`, `<M-e>`, `<C-a>`, `<C-e>`, `<M-'<'>`, `<M-'>'>`,
- ▶ Change/Delete/Replace: `<C-d>`, `<M-d-esc>`, `<M-d>`, `<C-kk>`, `<C-d'char'>`, `<Insert>`
- ▶ Copy/Paste: `<C-space>`, `<C-y>`, `<C-_>`, `<M-w>`, `<C-aky>`
- ▶ Search/Replace: `<C-s enter>`, `<C-s>`, `<C-r>`, `<M-x, 'replace-string'<CR>'srchstr'<CR>'replacement'<CR>`
- ▶ Save/Quit: `<C-xs>`, `<C-xw>`, `<C-xc, 'n', 'yes'<CR>>`
- ▶ Cheat sheet: [vim.pdf](#)
- ▶ Movement: `<h>`, `<j>`, `<k>`, `<l>`, ``, `<e>`, `<0>`, `<$>`, `<gg>`, `<G>`
- ▶ Change/Delete/Replace: `<x>`, `<cw>`, `<dw>`, `<dd>`, `<r>`, `<R>`
- ▶ Copy/Paste: `<v>`, `<P>`, `<u>`, `<y>`, `<yy>`
- ▶ Search/Replace: `</>`, `<n>`, `<N>`, `<:%s/'regex'/'replacement'/g>`
- ▶ Save/Quit: `<:q>`, `<:w>`, `<:q!>`

Emacs - Control Keys
C=Ctrl and M=Meta (Alt)



Vim - Modal
Cmd, Insert, and Visual





Mission Possible: Editing Files with Emacs and Vim

- Someone has corrupted Edgar Allen Poe's poem, "The Raven." Your mission, should you decide to accept it, is to repair the damage with `emacs` or `vim` and then confirm with the "diff" command. Hint: Also use `diff` to find corruption.
- `emacs -nw bad-the-raven.txt`
or
- `vim bad-the-raven.txt`
- After editing and saving your file, confirm you work with:
 - `diff bad-the-raven.txt good-the-raven.txt`



Finis 🐧



Unit-II

Overview of Linux



Dr. M. Durairaj

Associate Professor

School of Computer Science, Engineering and Applications

Bharathidasan University

Credits



□ Cleveland Linux Users' Group

- Introduction to Linux (Jeff Gilton & Jim Weirich)

□ IBM

- An Introduction to Linux (Al Henderson)
- Why Linux is storming the market (Jonathan Prial)

□ Ivan Bowman

- Conceptual software architecture of the Linux kernel

Contents



- A quick guide to Linux
 - Background
 - Using Linux
 - S/390 Specifics
- Linux in the Marketplace
- Commercial Linux Applications
- Additional Resources

What is Linux



- A fully-networked 32/64-Bit Unix-like Operating System
 - Unix Tools Like sed, awk, and grep (explained later)
 - Compilers Like C, C++, Fortran, Smalltalk, Ada
 - Network Tools Like telnet, ftp, ping, traceroute
- Multi-user, Multitasking, Multiprocessor
- Has the X Windows GUI
- Coexists with other Operating Systems
- Runs on multiple platforms
- Includes the Source Code

Where did it come from?



- Linus Torvalds created it
 - with assistance from programmers around the world
 - first posted on Internet in 1991
- Linux 1.0 in 1994; 2.2 in 1999
- Today used on 7-10 million computers
 - with 1000's of programmers working to enhance it

Open Source Software



- When programmers on the Internet can read, redistribute, and modify the source for a piece of software, **it evolves**
- People improve it, people adapt it, people fix bugs. And this can happen at a speed that, compared to conventional software development, seems **astounding**

How do you get it?



- Download it from the Internet
- From a “Distribution” (e.g. [RedHat](#))
 - Linux kernel
 - X Windows system and GUI
 - Web, e-mail, FTP servers
 - Installation & configuration support
 - 3rd party apps
 - Hardware support

Why is it significant?



- Growing popularity
- Powerful
 - Runs on multiple hardware platforms
 - Users like its speed and stability
 - No requirement for latest hardware
- It's "free"
 - Licensed under GPL
 - Vendors are distributors who package Linux

Linux/390

Using it



Powered by **S/390**

Logging In



- Connect to the Linux system using telnet:
 - vt100, vt220, vt320
 - ansi
 - tty
 - X-windows
- Able to login more than once with same user
- No 'MW' problems!

Logging In



- Before you can use it you must login by specifying your account and password:

```
Linux 2.2.13 (penguinvm.princeton.edu) (tty1)
penguinvm login: neale
Password: ←
Last login: Tue Jan  4 10:13:13 from
linuxtcp.princeton.edu
[neale@penguinvm neale]$
```

Rule Number 1



- Do not login as root unless you have to
- root is the system superuser (the “maint” of Linux but more “dangerous”)
 - Normal protection mechanisms can be overridden
 - Careless use can cause damage
 - Has access to everything by default
- root is the only user defined when you install
 - First thing is to change root’s password
 - The second job is to define “normal” users for everyday use

Creating a new user



- Use the useradd command
- Use the passwd command to set password
- Try it... logon as root

```
[root@penguinvm]# useradd scully
[root@penguinvm]# passwd scully
Changing password for user scully
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated
successfully
[root@penguinvm]#
```

Adding a new user



- Limits on users can be controlled by
 - Quotas
 - ulimit command
- Authority levels for a user controlled by group membership

Users and Groups

- Users are identified by user identifications (UIDs), each of which is associated with an integer in the range of 0 to 4 294 967 295 (X'FFFFFFFF'). Users with UID=0 are given superuser privileges.
- Users are placed in groups, identified by group identifications (GIDs). Each GID is associated with an integer in the range from 0 to 4 294 967 295
- Let the system assign UID to avoid duplicates
- Use id to display your user and group information

```
uid=500 (neale) gid=500 (neale) groups=500 (neale), 3 (sys), 4 (adm)
```


Users and Groups



- Groups define functional areas/responsibilities
- They allow a collection of users to share files
- A user can belong to multiple groups
- You can see what groups you belong to using the groups command:

```
neale sys adm
```

Typical Group Setup



sys

bin

adm

staff

Using the new user



- Now logoff using the exit command
- login as the new user

```
Linux 2.2.13 (penguinvm.princeton.edu) (tty2)

penguinvm login: scully
Password:
[scully@penguinvm scully]$
```

You need help?



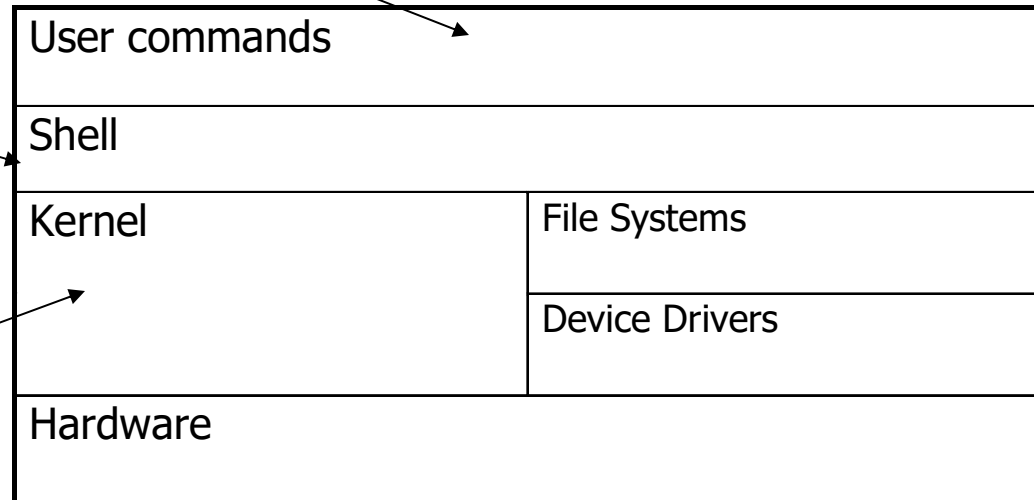
- The Linux equivalent of HELP is man (manual)
 - Use `man -k <keyword>` to find all commands with that keyword
 - Use `man <command>` to display help for that command
 - Output is presented a page at a time. Use `b` for to scroll backward, `f` or a space to scroll forward and `q` to quit

The Linux System

User commands includes executable programs and scripts

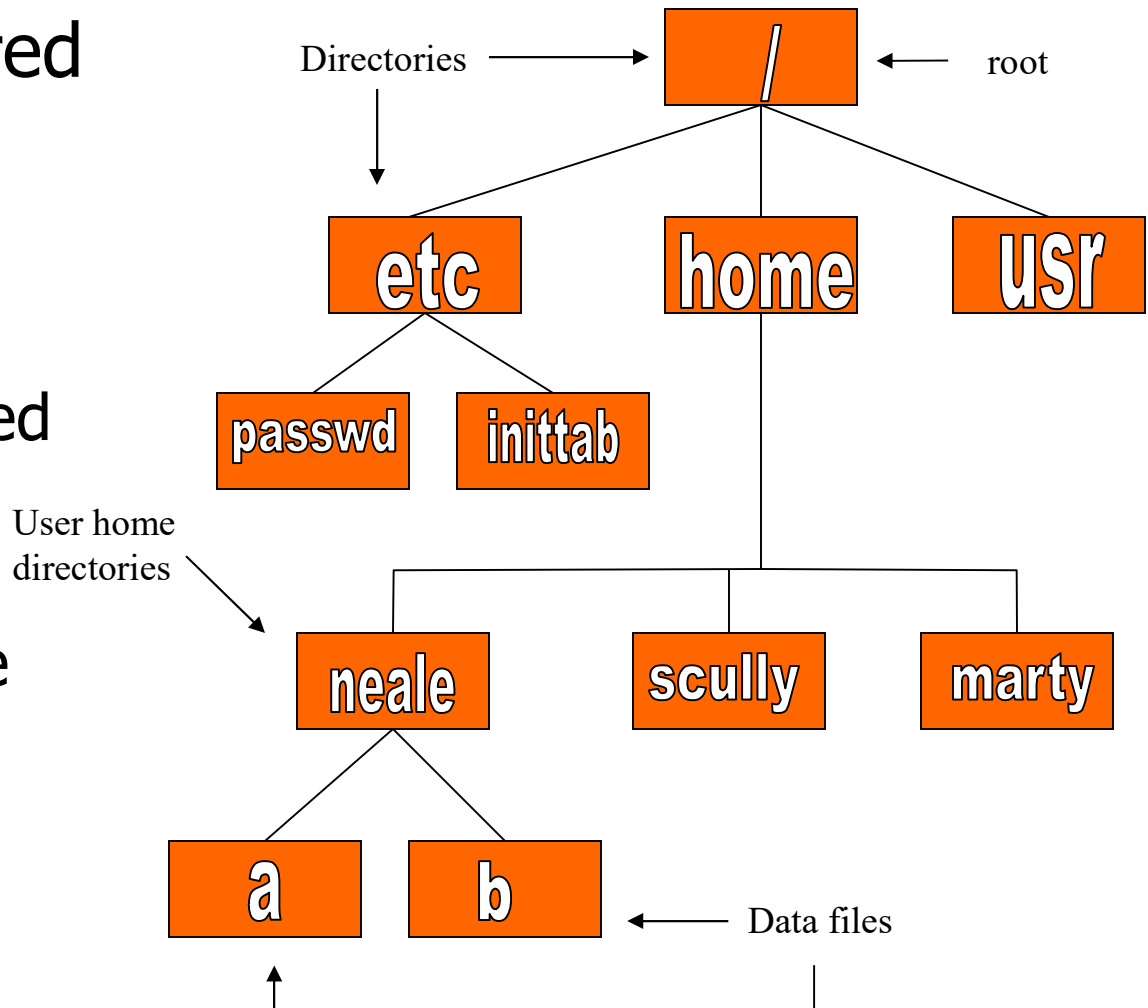
The shell interprets user commands. It is responsible for finding the commands and starting their execution. Several different shells are available. Bash is popular,

The kernel manages the hardware resources for the rest of the system.



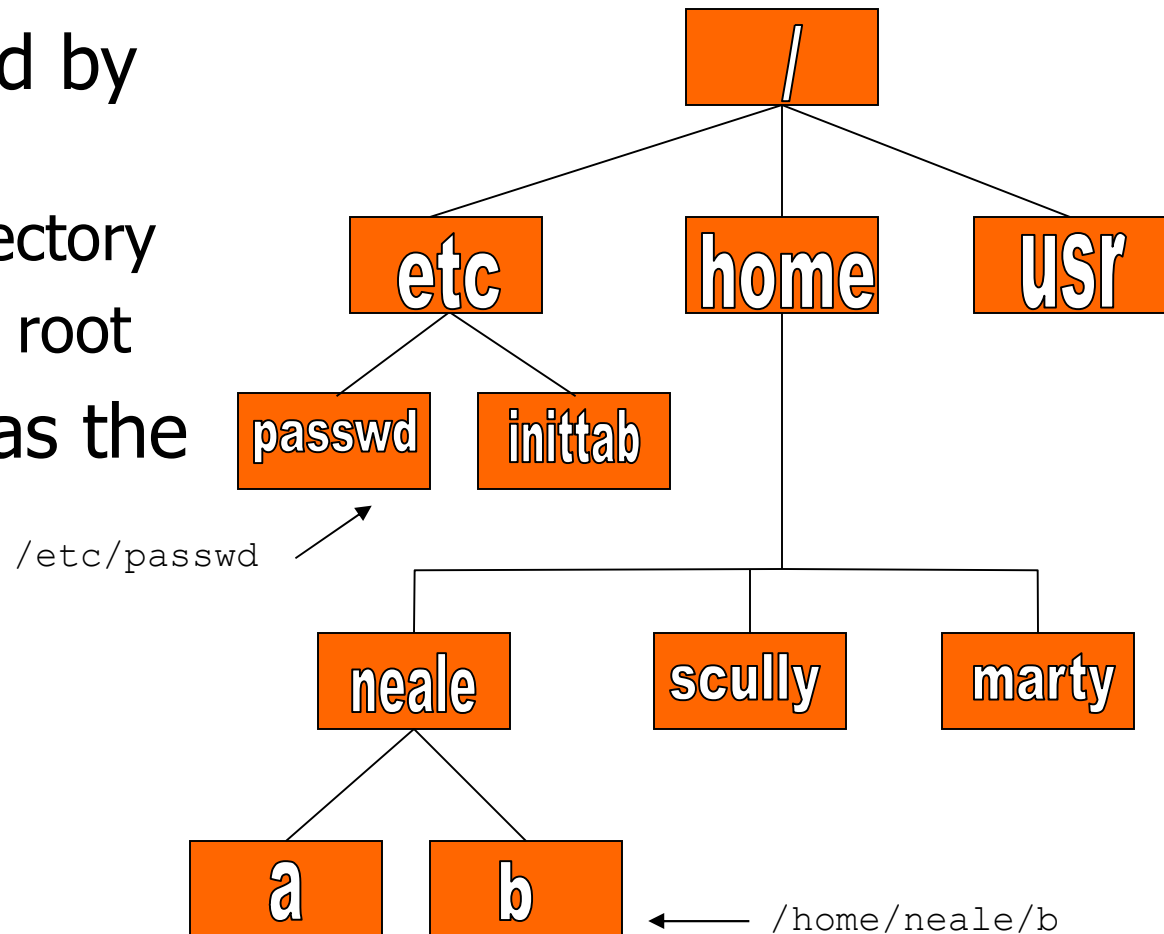
Linux File System Basics

- Linux files are stored in a single rooted, hierarchical file system
 - Data files are stored in directories (folders)
 - Directories may be nested as deep as needed



Naming Files

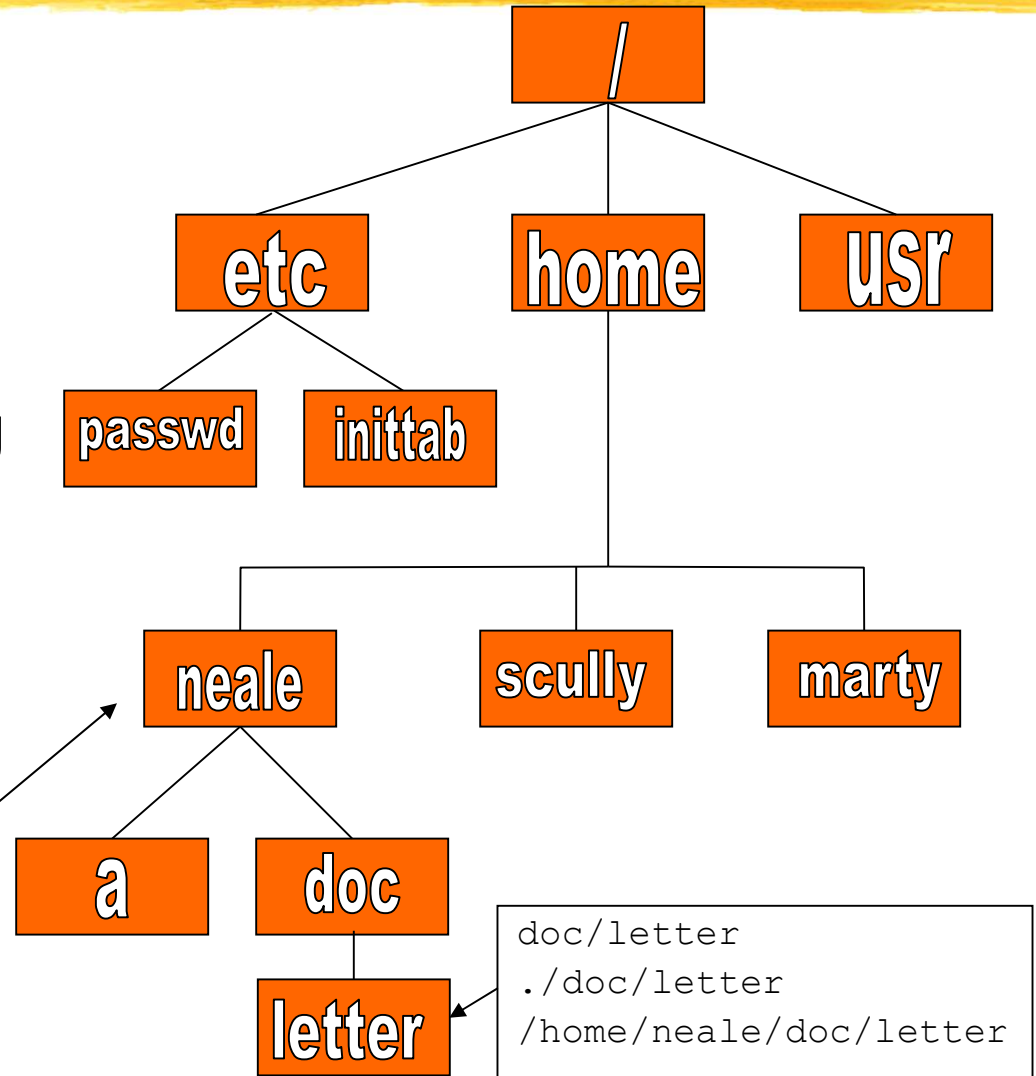
- Files are named by
 - naming each containing directory
 - starting at the root
- This is known as the *pathname*



The Current Directory

- One directory is designated the *current working directory*
 - if you omit the leading / then path name is relative to the current working directory
 - Use `pwd` to find out where you are

Current working directory



Some Special File Names



□ Some file names are special:

- / The root directory (not to be confused with the root user)
- . The current directory
- .. The parent (previous) directory
- ~ My home directory

□ Examples:

- ./a same as a
- ../jane/x go up one level then look in directory jane for x

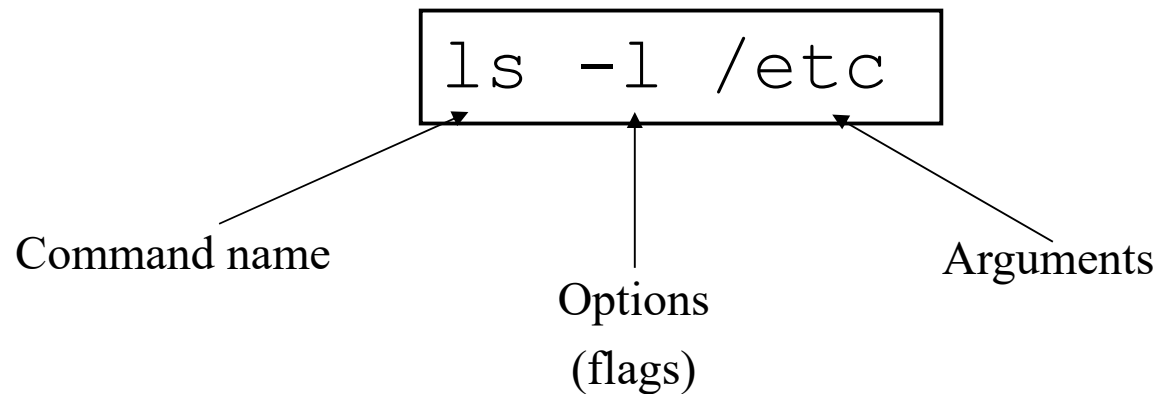
Special Files



- `/home` - all users' home directories are stored here
- `/bin, /usr/bin` - system commands
- `/sbin, /usr/sbin` - commands used by sysadmins
- `/etc` - all sorts of configuration files
- `/var` - logs, spool directories etc.
- `/dev` - device files
- `/proc` - special system files

Linux Command Basics

- To execute a command, type its name and arguments at the command line



Standard Files



- UNIX concept of “standard files”
 - standard input (where a command gets its input) - default is the terminal
 - standard output (where a command writes its output) - default is the terminal
 - standard error (where a command writes error messages) - default is the terminal

Redirecting Output

- The output of a command may be sent (piped) to a file:

```
ls -l >output
```

“>” is used to specify
the output file

Redirecting Input

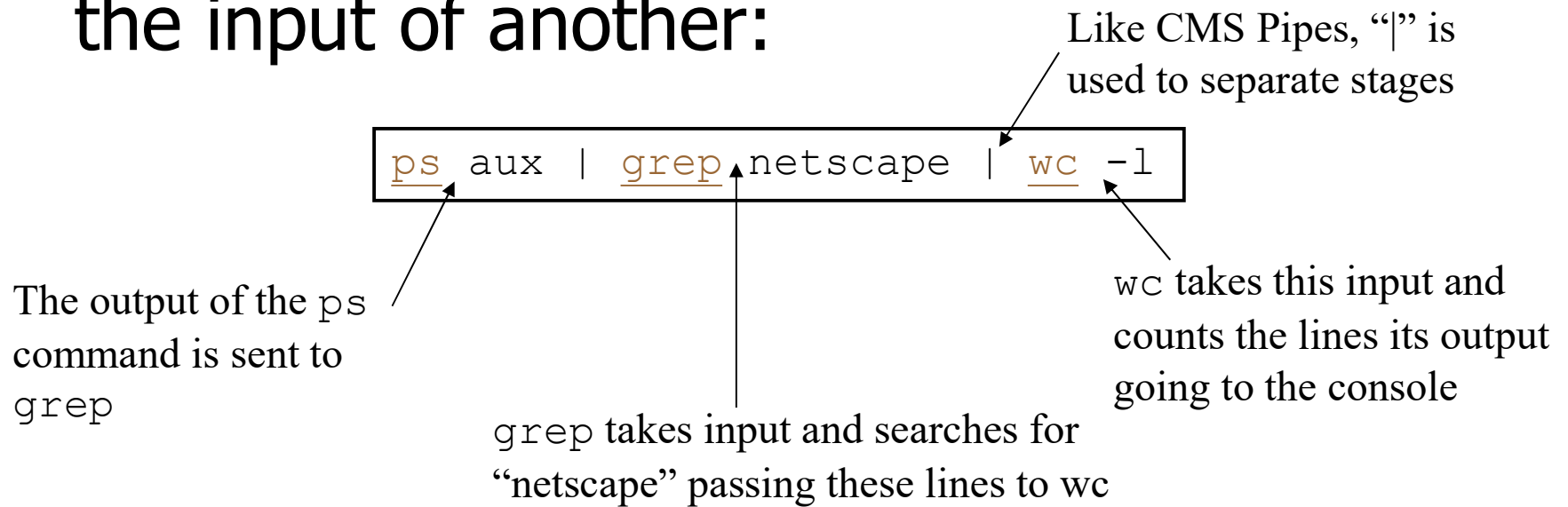
- The input of a command may come (be piped) from a file:

```
wc <input
```

“<” is used to specify
the input file

Connecting commands with Pipes

- Not as powerful as CMS Pipes but the same principle
- The output of one command can become the input of another:



Command Options



- Command options allow you to control a command to a certain degree
- Conventions:
 - Usually being with a single dash and are a single letter (“-l”)
 - Sometimes have double dashes followed by a keyword (“--help”)
 - Sometimes follow no pattern at all

Common Commands



- pwd - print (display) the working directory
- cd <*dir*> - change the current working directory to *dir*
- ls - list the files in the current working directory
- ls -l - list the files in the current working directory in long format

File Commands



- cp *<fromfile>* *<tofile>*
 - Copy from the *<fromfile>* to the *<tofile>*
- mv *<fromfile>* *<tofile>*
 - Move/rename the *<fromfile>* to the *<tofile>*
- rm *<file>*
 - Remove the file named *<file>*
- mkdir *<newdir>*
 - Make a new directory called *<newdir>*
- rmdir *<dir>*
 - Remove an (empty) directory

More Commands



- who
 - List who is currently logged on to the system
- whoami
 - Report what user you are logged on as
- ps
 - List your processes on the system
- ps aux
 - List all the processes on the system
- echo "*A string to be echoed*"
 - Echo a string (or list of arguments) to the terminal

More Commands



□ alias - used to tailor commands:

□ `alias erase=rm`

□ `alias grep="grep -i"`

□ ar - Maintain archive libraries: a collection of files (usually object files which may be linked to a program, like a CMS TXTLIB)

```
ar -t libgdbm.a
    .SYMDEF
dbmopen.o
```

More Commands



- awk - a file processing language that is well suited to data manipulation and retrieval of information from text files
- chown - sets the user ID (UID) to owner for the files and directories named by pathname arguments. This command is useful when from test to production

```
chown -R apache:httpd /usr/local/apache
```

More Commands



- diff - attempts to determine the minimal set of changes needed to convert a file specified by the first argument into the file specified by the second argument
- find - Searches a given file hierarchy specified by path, finding files that match the criteria given by expression

More Commands

- grep - Searches files for one or more pattern arguments. It does plain string, basic regular expression, and extended regular expression searching

```
find ./ -name "* .c" | xargs grep -i "fork"
```

In this example, we look for files with an extension "c" (that is, C source files). The filenames we find are passed to the xargs command which takes these names and constructs a command line of the form: `grep -i fork <file.1>...<file.n>`. This command will search the files for the occurrence of the string "fork". The "-i" flag makes the search case insensitive.

More Commands

- kill - sends a signal to a process or process group
- You can only kill your own processes unless you are root

```
UID          PID    PPID    C  STIME TTY          TIME CMD
root         6715   6692    2  14:34 ttyp0        00:00:00 sleep 10h
root         6716   6692    0  14:34 ttyp0        00:00:00 ps -ef
[root@penguinvm log]# kill 6715
[1]+  Terminated                  sleep 10h
```


More Commands



- make - helps you manage projects containing a set of interdependent files (e.g. a program with many source and object files; a document built from source files; macro files)
- `make` keeps all such files up to date with one another: If one file changes, `make` updates all the other files that depend on the changed file
- Roughly the equivalent of VMFBLD

More Commands



- sed - applies a set of editing subcommands contained in a script to each argument input file

```
find ./ -name "*.c,v" | sed 's/,v//g' | xargs grep "PATH"
```

This `find`s all files in the current and subsequent directories with an extension of `c,v`. `sed` then strips the `,v` off the results of the `find` command. `xargs` then uses the results of `sed` and builds a `grep` command which searches for occurrences of the word `PATH` in the C source files.

More Commands

□ tar - manipulates archives

- An archive is a single file that contains the complete contents of a set of other files; an archive preserves the directory hierarchy that contained the original files. Similar to a VMARC file

```
tar -tzf imap-4.7.tar.gz
imap-4.7/
imap-4.7/src/
imap-4.7/src/c-client/
imap-4.7/src/c-client/env.h
imap-4.7/src/c-client/fs.h
```

Shells



- An interface between the Linux system and the user
- Used to call commands and programs
- An interpreter
- Powerful programming language
 - “Shell scripts” = .bat .cmd EXEC REXX
- Many available (bsh; ksh; csh; bash; tcsh)

Another definition of a Shell



- A shell is any program that takes input from the user, translates it into instructions that the operating system can understand, and conveys the operating system's output back to the user.
 - i.e. Any User Interface
 - Character Based v Graphics Based

Why Do I Care About The Shell?



□ Shell is Not Integral Part of OS

- UNIX Among First to Separate
- Compare to MS-DOS, Mac, Win95, VM/CMS
- GUI is NOT Required
- Default Shell Can Be Configured
 - `chsh -s /bin/bash`
 - `/etc/passwd`
- Helps To Customize Environment

Shell Scripts



```
#!/bin/bash
while
true
do
    cat somefile > /dev/null
    echo .
done
```

```
/* */
do forever
    'PIPE < SOME FILE | hole'
    say `.`
end
```

Switching Users



- su *<accountname>*

- switch user accounts. You will be prompted for a password. When this command completes, you will be logged into the new account. Type `exit` to return to the previous account

- `su`

- Switch to the root user account. Do not do this lightly

- **Note:** The root user does not need to enter a password when switching users. It may become any user desired. This is part of the power of the root account.

Environment Variables



- Environment variables are global settings that control the function of the shell and other Linux programs. They are sometimes referred to global shell variables.
- Setting:
 - `VAR=/home/fred/doc`
 - `export TERM=ansi`
 - `SYSTEMNAME=`uname -n``
- Similar to GLOBALV SET ... in CMS

Environment Variables



- Using Environment Variables:
 - `echo $VAR`
 - `cd $VAR`
 - `cd $HOME`
 - `echo "You are running on $SYSTEMNAME"`
- Displaying - use the following commands:
 - `set` (displays local & env. Vars)
 - `export`
- Vars can be retrieved by a script or a program

Some Important Environment Variables



□ HOME

- Your home directory (often be abbreviated as "~")

□ TERM

- The type of terminal you are running (for example vt100, xterm, and ansi)

□ PWD

- Current working directory

□ PATH

- List of directories to search for commands

PATH Environment Variable

- Controls where commands are found
 - PATH is a list of directory pathnames separated by colons. For example:
 - `PATH=/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/home/scully/bin`
 - If a command does not contain a slash, the shell tries finding the command in each directory in PATH. The first match is the command that will run

PATH Environment Variable



- Similar to setting the CMS search order
- Usually set in `/etc/profile` (like the SYSPROF EXEC)
- Often modified in `~/profile` (like the PROFILE EXEC)

File Permissions



- Every file
 - Is owned by someone
 - Belongs to a group
 - Has certain access permissions for owner, group, and others
 - Default permissions determined by umask

File Permissions



- Every user:
 - Has a *uid* (login name), *gid* (login group) and membership of a "groups" list:
 - The *uid* is who you are (name and number)
 - The *gid* is your initial "login group" you normally belong to
 - The *groups list* is the file groups you can access via group permissions

File Permissions



- Linux provides three kinds of permissions:
 - Read - users with read permission may read the file or list the directory
 - Write - users with write permission may write to the file or new files to the directory
 - Execute - users with execute permission may execute the file or lookup a specific file within a directory

File Permissions

- The long version of a file listing (`ls -l`) will display the file permissions:

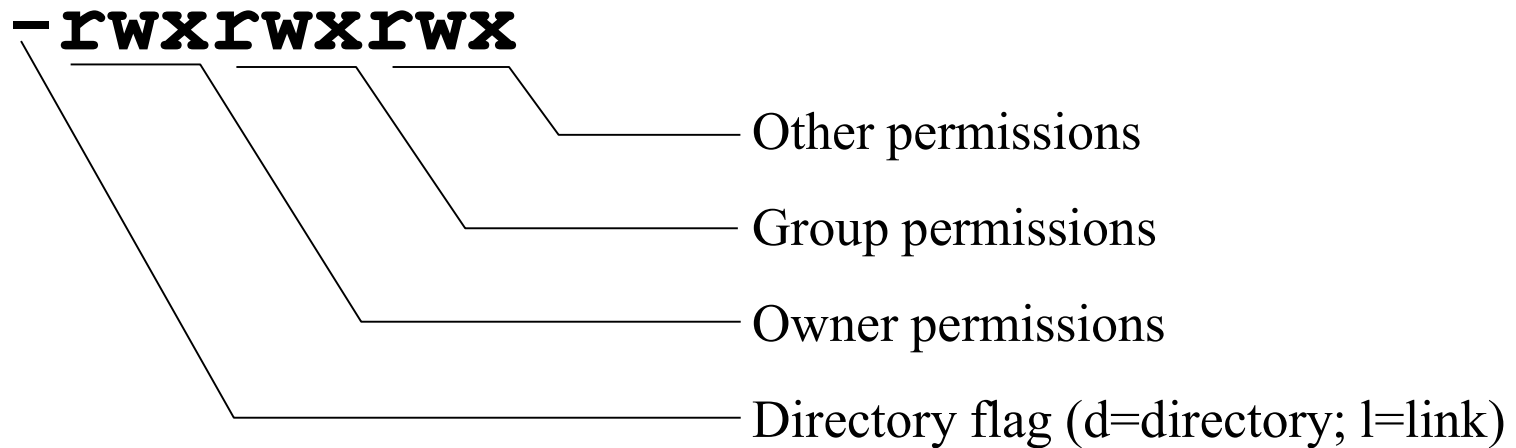
```
-rwxrwxr-x  1 rvdheij  rvdheij    5224 Dec 30 03:22 hello
-rw-rw-r--  1 rvdheij  rvdheij     221 Dec 30 03:59 hello.c
-rw-rw-r--  1 rvdheij  rvdheij    1514 Dec 30 03:59 hello.s
drwxrwxr-x  7 rvdheij  rvdheij    1024 Dec 31 14:52 posixuft
```

Permissions

Owner

Group

Interpreting File Permissions



Changing File Permissions

- Use the chmod command to change file permissions
 - The permissions are encoded as an octal number

```
chmod 755 file # Owner=rwx Group=r-x Other=r-x
chmod 500 file2 # Owner=r-x Group=--- Other=---
chmod 644 file3 # Owner=rw- Group=r-- Other=r--

chmod +x file # Add execute permission to file for all
chmod o-r file # Remove read permission for others
chmod a+w file # Add write permission for everyone
```

Links?



- Links are references to files (aliases)
- Two forms:
 - Hard
 - Symbolic
 - Can point to files on different physical devices
 - Delete of original leaves link
 - Delete of link leaves original
 - Can be created for directories
- Create using ln command

Editors



- People are fanatical about their editor
- Several choices available:
 - vi Standard UNIX editor
 - the XEDIT-like editor
 - xedit X windows text editor
 - emacs Extensible, Customizable Self-Documenting Display Editor
 - pico Simple display-oriented text editor
 - nedit X windows Motif text editor

Linux Device Handling

- Devices are the way linux talks to the world
- Devices are special files in the `/dev` directory (try `ls /dev`)

<code>/dev/ttyx</code>	TTY devices
<code>/dev/hdb</code>	IDE hard drive
<code>/dev/hdb1</code>	Partition 1 on the IDE hard drive
<code>/dev/mnda</code>	VM Minidisk
<code>/dev/dda</code>	Channel Attached DASD
<code>/dev/dda1</code>	Partition 1 on DASD
<code>/dev/null</code>	The null device ("hole")
<code>/dev/zero</code>	An endless stream of zeroes
<code>/dev/mouse</code>	Link to mouse (not <code>/390</code>)

Devices and Drivers

- Each `/dev` file has a major and minor number
 - Major defines the device type
 - Minor defines device within that type
 - Drivers register a device type

<code>brw-r--r--</code>	<code>1</code>	<code>root</code>	<code>root</code>	<code>64,</code>	<code>0</code>	<code>Jun</code>	<code>1</code>	<code>1999</code>	<code>/dev/mnda</code>
<code>crw-r--r--</code>	<code>1</code>	<code>root</code>	<code>root</code>	<code>5,</code>	<code>0</code>	<code>Jan</code>	<code>5</code>	<code>09:18</code>	<code>/dev/tty</code>

Device Type:
b - block
c - character

Major no.

Minor no.

Special Files - /proc

- Information about internal Linux processes are accessible to users via the `/proc` file system (in memory)

<code>/proc/cpuinfo</code>	CPU Information
<code>/proc/interrupts</code>	Interrupt usage
<code>/proc/version</code>	Kernel version
<code>/proc/modules</code>	Active modules

```
cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 1
bogomips per cpu: 86.83
processor 0: version = FF, identification = 045226, machine = 9672
```


File Systems



- Linux supports many different types
- Most commonly, ext2fs
 - Filenames of 255 characters
 - File sizes up to 2GB
 - Theoretical limit 4TB
- Derived from extfs
- Highly reliable and high performer

File Systems



□ Other file systems:

- sysv - SCO/Xenix
- ufs - SunOS/BSD
- vfat - Win9x
- msdos - MS-DOS/Win
- umsdos - Linux/DOS
- ntfs - WinNT (r/o)
- hpfs - OS/2 (r/o)

□ Other File systems:

- iso9660 (CD-ROM)
- nfs - NFS
- coda - NFS-like
- ncp - Novell
- smb - LANManager
etc

File Systems



□ mount

- Mounts a file system that lives on a device to the main file tree
- Start at Root file system
 - Mount to root
 - Mount to points currently mounted to root
- `/etc/fstab` used to establish boot time mounting

Virtual File System



- VFS is designed to present a consistent view of data as stored on hardware
- Almost all hardware devices are represented using a generic interface
- VFS goes further, allowing the sysadmin to mount any of a set of logical file systems on any physical device

Virtual File System

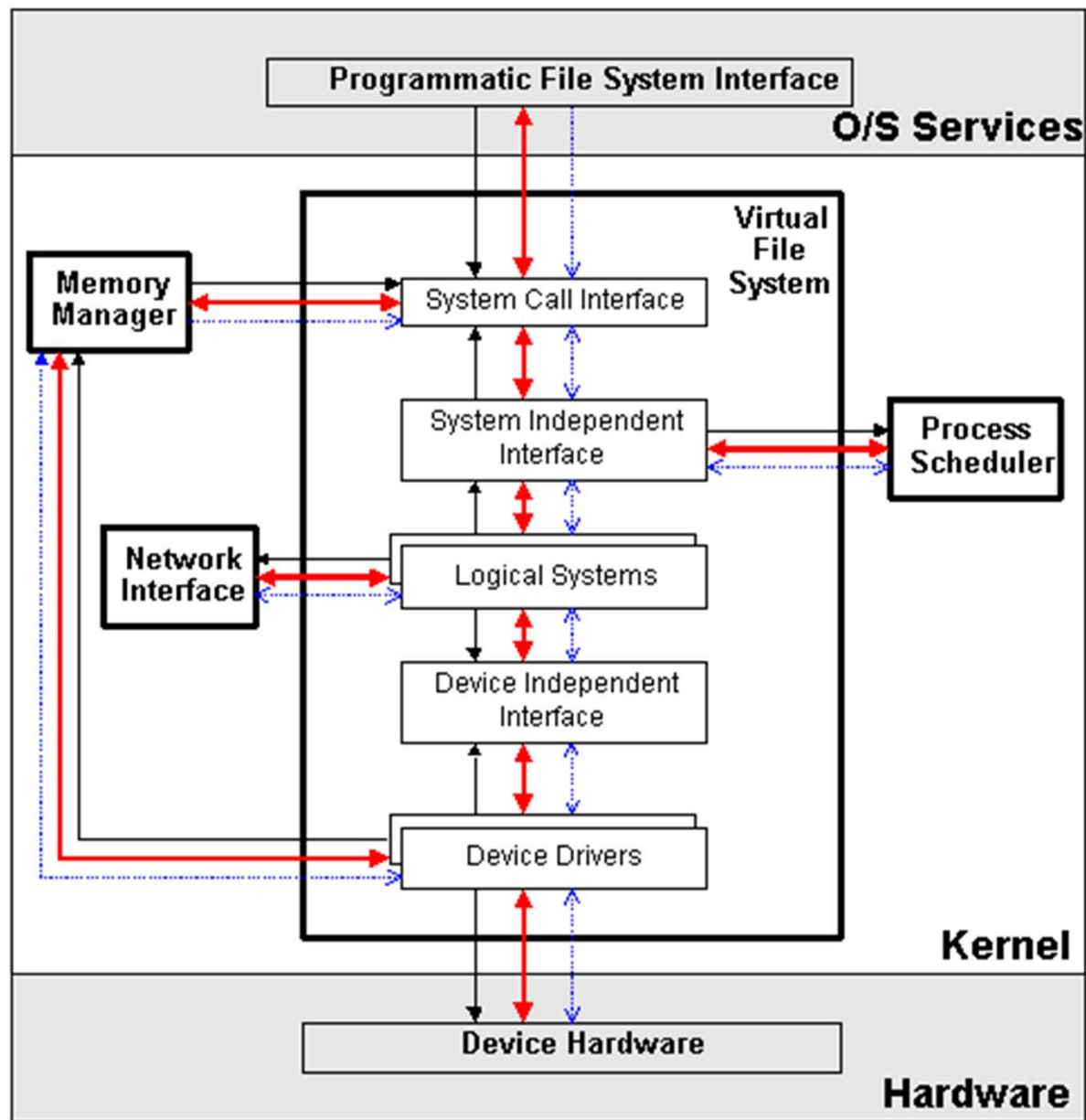


- Logical file systems promote compatibility with other operating system standards permitting developers to implement file systems with different policies
- VFS abstracts details of physical device and logical file system allowing processes to access files using a common interface, without knowing what physical or logical system the file resides on

Virtual File System



- Analogous to CMS:
 - SFS
 - Minidisks
- Two different designs
- Common/transparent access



Processes



- Processes are created in a hierarchical structure whose depth is limited only by the virtual memory available to the virtual machine
- A process may control the execution of any of its descendants by suspending or resuming it, altering its relative priority, or even terminating it
- Termination of a process by default causes termination of all its descendants; termination of the root process causes termination of the session
- Linux assigns a *process ID* (PID) to the process

Processes



□ Foreground

- When a command is executed from the prompt and runs to completion at which time the prompt returns is said to run in the foreground

□ Background

- When a command is executed from the prompt with the token "&" at the end of the command line, the prompt immediately returns while the command continues is said to run in the background

Processes

□ Daemons

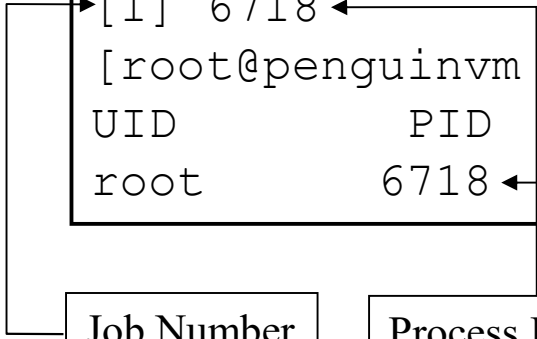
- Background processes for system administration are referred to as “daemons”
- These processes are usually started during the boot process
- The processes are not assigned any terminals

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	5	1	0	1999	?	00:00:14	[kswapd]
bin	254	1	0	1999	?	00:00:00	[portmap]
root	307	1	0	1999	?	00:00:23	syslogd -m 0
root	350	1	0	1999	?	00:00:34	httpd

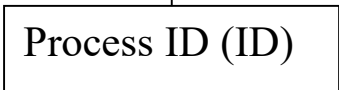
Processes

& causes process to be run in "background"

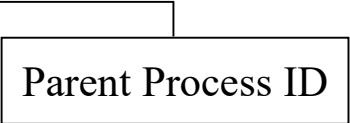
```
[root@penguinvm log]# sleep 10h &
[1] 6718
[root@penguinvm log]# ps -ef
UID          PID    PPID  C  STIME TTY          TIME CMD
root         6718   6692  0  14:49 ttyp0      00:00:00 sleep 10h
```



Job Number



Process ID (ID)



Parent Process ID

Processes - UID & GID



□ Real UID

- At process creation, the real UID identifies the user who has created the process

□ Real GID

- At process creation, the real GID identifies the current connect group of the user for which the process was created

Processes - UID & GID



□ Effective UID

- The effective UID is used to determine owner access privileges of a process.
- Normally the same as the real UID. It is possible for a program to have a special flag set that, when this program is executed, changes the effective UID of the process to the UID of the owner of the program.
- A program with this special flag set is said to be a set-user-ID program (SUID). This feature provides additional permissions to users while the SUID program is being executed.

Processes - UID & GID



□ Effective GID

- Each process also has an effective group
- The effective GID is used to determine group access privileges of a process
- Normally the same as the real GID. A program can have a special flag set that, when this program is executed, changes the effective GID of the process to the GID of the owner of this program
- A program with this special flag set is said to be a set-group-ID program (SGID). Like the SUID feature, this provides additional permission to users while the set-group-ID program is being executed

Processes - Process Groups



- Each process belongs to a process group
- A *process group* is a collection of one or more processes
- Each process group has a unique process group ID
- It is possible to send a signal to every process in the group just by sending the signal to the process group leader
- Each time the shell creates a process to run an application, the process is placed into a new process group
- When an application spawns new processes, these are members of the same process group as the parent

Processes - PID



□ PID

- A process ID is a unique identifier assigned to a process while it runs
- Each time you run a process, it has a different PID (it takes a long time for a PID to be reused by the system)
- You can use the PID to track the status of a process with the ps command or the jobs command, or to end a process with the kill command

Processes - PGID



□ PGID

- Each process in a process group shares a process group ID (PGID), which is the same as the PID of the first process in the process group
- This ID is used for signaling-related processes
- If a command starts just one process, its PID and PGID are the same

Processes - PPID



□ PPID

- A process that creates a new process is called a *parent process*; the new process is called a *child process*
- The parent process (PPID) becomes associated with the new child process when it is created
- The PPID is not used for job control

Security Guidelines



- Take Care With Passwords
 - Use good ones (motherhood statement)
 - Don't Use Real Words
 - Make Sure They Are Not Easily Guessed
 - Use Combinations Of Upper and Lower Case, Numbers, Punctuation
 - One Method: Take first letter of a sentence or book title, insert numbers and punctuation.

Security Guidelines



- Take care of passwords (continued)
 - Use Shadow Passwords
 - Allows encrypted passwords to be in a file that is not world readable
 - Use Password Aging
 - Requires shadow passwords

Security Guidelines



□ Restrict Superuser Access

□ Restrict where root can log in from

- `/etc/securetty` restricts root access to devices listed

□ Use wheel group to restrict who can su to root

- Put users who can `su` to root in wheel group in `/etc/group` file.

Security Guidelines




- Use groups to allow access to files that must be shared
 - Otherwise users will set world permission
- Be careful with SUID and SGID
 - Avoid setting executables to SUID root
 - Wrap SUID root wrapper around programs if they must be run SUID root
 - Create special accounts for programs that must run with higher permissions

Security - Important Files




```
/etc/passwd - password file
/etc/shpasswd - shadow password file
/etc/group - lists groups and users contained in groups
/etc/services - lists network services and their ports
/etc/ftpusers - contains list of accounts that cannot use ftp
/etc/hosts.equiv - generic list of remote users
~/.rhosts - list of remote users for a specific account
/etc/hosts - host definition list
/etc/hosts.lpd - hosts who can use remote printing
/etc/hosts.allow - lists services that remote users are allowed to use
/etc/hosts.deny - lists services that remote users are not allowed to use
/etc/nologin - no login message that also disables logins
/etc/securetty - lists legal terminals for root to login from
/etc/exports - lists locations that can be remotely accessed via NFS
/etc/syslog.conf - configures the syslog facility
/etc/inetd.conf - configures inetd
```

Linux/390 Specifics




- An ASCII implementation
- Adds a layer of abstraction to I/O
 - Channel based v IRQ based
- Support for ECKD using SSCH
- Support for VM minidisks (ECKD, CKD, FBA, VDISK)

Linux/390 Specifics



- Runs natively, in LPAR, or under VM/ESA
- Uses relative instructions: G2, P/390, R/390 or better
- Will use hardware IEEE FP or will emulate
- Network drivers for CTCA/ESCON, OSA-2, and IUCV (VM only)
- 3215 emulation for virtual console
- Hardware console driver (HMC)

Linux/390 Specifics



- GNU tools ported
 - C/C++ compiler ([gcc-2.95.1](#))
 - Assembler and linker ([binutils-2.9.1](#))
- Packages “ported”:
 - [Regina](#); [THE](#); [UFT](#); [X11](#); [OpenLDAP](#); [IMAP](#); [Sendmail](#); [Bind](#); [RPM](#); [Samba 2.0.6](#); [Apache](#); [Perl](#)

Linux in the Business World



Issues and observations

Linux's place in the market



- The business world is interested in:
 - Efficiency and effectiveness
 - Networked economy
 - Network-based businesses

Linux's place in the market



- The world is heterogeneous
 - 90% of Fortune 1000 companies use 3 or more Operating Systems
- The demands of e-business
 - Integrates with existing investments
 - Supports any client
 - Applications built/deployed independent of client
 - 24 x 7

Linux's place in the market



- Importance of the application model
 - Server-centric and based on standards that span multiple platforms
 - Leverage core business systems and scale to meet unpredictable demands
 - Quick to deploy, easy to use and manage

Linux's place in the market



- ISVs which have made Linux announcements:
 - BEA; Novell; SAP; Informix; Oracle, IBM; HP; CA; ApplixWare; Star; Corel; Cygnus; MetroWerks; ObjectShare; Inprise
- Media spotlight:
 - CNN; PCWorld; PCWeek; InternetWeek

Linux's place in the market



- Early commercial users
 - Cendant Corporation - 4000 hotels
 - Burlington Coat Factory - back office functions
 - Northwest Airlines - 23 flight simulators
- Intel announcement January 5 2000
 - New web appliances to run Linux
 - At the insistence of customers (e.g. NEC)

Linux's place in the market



□ Impacts:

□ Applications:

- Webservers (65%)
- WebInfrastructure (mail, DNS) (15%)
- File/Print (15%)
- DB & DB Applications (2%)

□ Observations

- Linux/Apache share of Web serving high
- Autonomous departments
- Many SMB and small ISP
- CIOs discovering they have Linux running somewhere
- Strong mindshare among developers

Linux's place in the market



□ Linux's appeal

- Embraces new generation of web-based apps
- Player in the heterogeneous e-business world
- Provides flexibility and choice of environment
- Open Source focuses on open standards

Linux's place in the market



- Challenges for growth
 - Products/Technologies/Offerings
 - Support services
 - ISV applications
 - Service providers
 - Trends
 - Movement to mainstream
 - Standards
 - Ease of use

IBM's focus on Linux



Services	Support offering; Curriculum
Software	Porting all key products to Linux
Hardware	Intel; RS/6000; S/390
Alliances	Partner with Caldera; Redhat; SuSe
Open Source	Support standards & contribute to bodies

IBM Software Announcements



- DB2 Universal Database
- Transarc AFS (distributed file system)
- On Demand Server
- Lotus Domino R5
- WebSphere
- Tivoli

Linux's place in the market



□ Summary

- Linux is viable in many key application areas
- Linux has moved from small technical projects to significant deployment
- IBM claims to be fully supportive of Linux
 - Part of their heterogeneous strategy
 - Open source supporter
 - Hardware, software, and service offerings

Linux



Available Commercial
Software

Website Development



- ASWedit, HTML editor
- Empress DataWEB
- EZ-EDIT
- LinkScan
- TalentSoft Web+ (WebPlus)
- VirtuFlex 1.1
- Visual prolog
- Web Crossing
- ThreadTrack
- WebTailor from Webthreads.

Databases



- c-tree Plus
- Empress
- Essentia
- FairCom Server
- INFORMIX-SE
- Just Logic/SQL
- KE Texpress
- Qddb
- Raima Database Manager++
- Empress Embedded RDBMS
- SOLID Server
- Velocis Database Server
- Yard SQL

Data Visualization and CAD



- IDL (Interactive Data Language)
- Megahedron
- Tecplot 7.0
- VariCAD
- VARKON
- XVScan

Development Tools



- ACUCOBOL-GT
- Amzi! Prolog & Logic Server
- Basmark QuickBASIC
- Critical Mass CM3
- Dynace
- Absoft Fortran 77
- Finesse
- ISE Eiffel
- EiffelBench
- C-Forge IDE
- IdeaFix
- j-tree
- KAI C++
- Khoros Pro 2.1

Development Tools



- MetaCard
- ObjectManual Rel 3.0
- Critical Mass Reactor
- Resource Standard Metrics
- r-tree
- sdoc (Source Documenter)
- SEDIT, S/REXX
- SNIFF+
- ST/X (Smalltalk/X)
- tdb (Tcl Debugger)
- tprof (Tcl Profiler)
- View Designer/X (VDX)
- XBasic
- XMove 4.0 for Linux

Emulation Tools



- Emulus
- Executor 2
- Wabi 2.2 for OpenLinux

Financial Software



- BB Stock Pro and BB Stock Tool
- TimeClock

Libraries



- FontScope
- INTERACTER
- Matrix<LIB> - C++ Math Matrix Library
- PKWARE Data Compression Library for Linux
- readyBase
- SIMLIB IG

Mathematics



- Maple V Release 4 - The Power Edition
- MATCOM and MATCOM MATH LIBRARY
- Mathematica 3.0
- MATLAB and Simulink

Multimedia



- Peter Lipa and his Journeys
- Lucka Vondrackova and her Journeys
- MpegTV Player 1.0
- Peter Nagy and his Journeys
- Xaudio

Network Servers



- Critical Angle X.500 Enabler
- DNEWS News Server
- Aventail Internet Policy Manager
- Aventail VPN
- WANPIPE
- Zeus Web Server

Office Tools



- Corel WordPerfect 8
- The American Heritage Dictionary Deluxe
- Applixware Office Suite
- D.M.S. Document Management System
- HotWire EasyFAX
- NExS, the Network Extensible Spreadsheet
- Axene Office
- Projector and Projector/Net
- The Virtual Office System
- Axene XAllWrite
- Axene Xclamation
- Axene XQuad

Text Processing



- Edith Pro for X11
- TeraSpell 97 for Emacs

System Administration



- Host Factory
- PerfectBACKUP+
- Venus

X Windows Related



- Accelerated-X Display Server
- BXwidgets
- BXwidgets/DB
- Laptop, Accelerated-X Display Server
- MaXimum cde Developer's Edition v1.0
- Multi-headed, Accelerated-X Display Server
- OpenGL, Accelerated-X Display Server
- OSF-Certified Motif

Other Software



- ABACUS 4
- BBBS
- Clustor
- FootPrints
- Aladdin Ghostscript
- Magician
- journyx WebTime
- LanSafe
- LjetMgr
- Synchronize/CyberScheduler

Additional Resources



- UNIX Systems Administrator Resources
 - <http://www.ugu.com/>
- Linux/390 Observations and Notes
 - <http://penguinvm.princeton.edu>
- Introduction to Linux
- Introduction to UNIX
- Linux/390 Installation
- Linux Administration Made Easy
 - <http://www.linuxninja.com/linux-admin/book1.html>
- Conceptual software architecture of the Linux kernel

Additional Resources



- <http://www.linux.org>
- <http://www.tux.org>
- <http://www.li.org>

Linux introduction



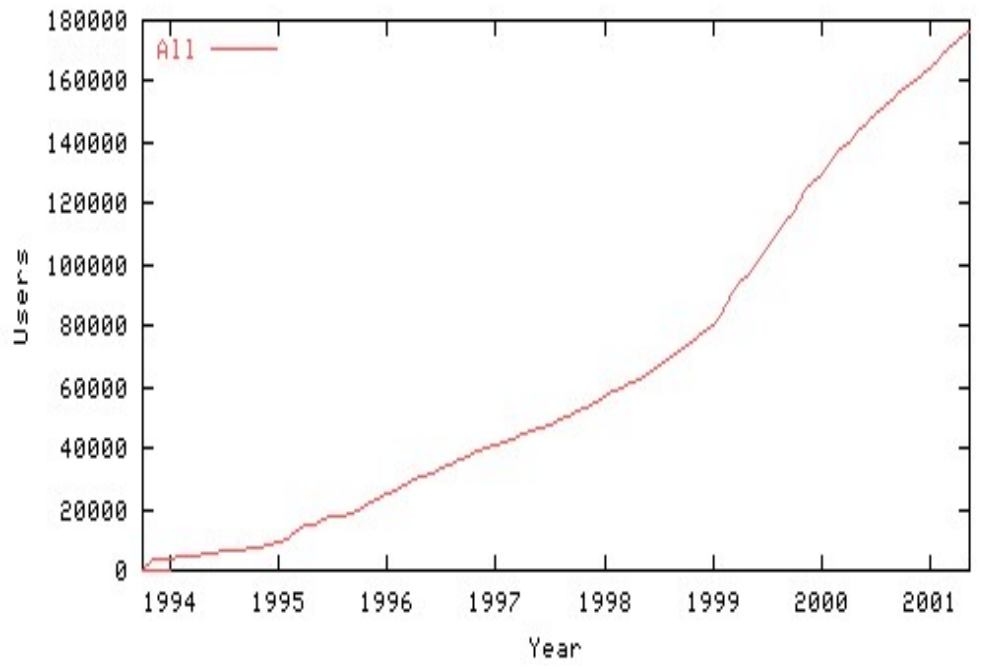
- Dinesh Gupta
- ICGEB, India

Linux

The Linux operating system (OS) was first coded by a Finnish computer programmer called **Linus Benedict Torvalds** in 1991, when he was just 21! He had got a new 386, and he found the existing DOS and UNIX too expensive and inadequate.



In those days, a UNIX-like tiny, free OS called **Minix** was extensively used for academic purposes. Since its source code was available, Linus decided to take Minix as a model.





Linux directories

- **/bin** System binaries, including the command shell
 - **/boot** Boot-up routines
- **/dev** Device files for all your peripherals
 - **/etc** System configuration files
 - **/home** User directories
 - **/lib** Shared libraries and modules
- **/lost+found** Lost-cluster files, recovered from a disk-check
 - **/mnt** Mounted file-systems
 - **/opt** Optional software
- **/proc** Kernel-processes pseudo file-system
 - **/root** Administrator's home directory
 - **/sbin** System administration binaries
 - **/usr** User-oriented software

/ bin
boot
dev
etc
home
lib
/ lost+found
misc
mnt
opt
proc
root
sbin
tmp

Why use Linux

- A Linux distribution has software worth thousands of dollars, for virtually no cost
- Linux operating system is **reliable, stable**, and **very powerful**
- Linux comes with a **complete development environment**, including compilers, toolkits, and scripting languages
- Linux comes with **networking** facilities, allowing you to share hardware
- Linux utilizes your memory, CPU, and other hardware to the fullest
- A wide variety of **commercial software** is also available
- Linux is very easily **upgradeable**
- Supports **multiple processors** as standard
- True **multitasking**. So many apps, all at once
- The GUIs are more powerful than Mac!

11/9/2024 3:57 PM

Why Linux in Bioinformatics ?

- One definition of bioinformatics is "the use of computers to analyze biological problems."
- As biological data sets have grown larger and biological problems have become more complex, the requirements for computing power have also grown.
- Computers that can provide this power generally use the Unix operating system - so you must learn Unix
- Linux/UNIX has powerful text processing tools which are highly suited to working with sequence data
 - While many bioinformatics tools have Web interfaces, many more are available via the UNIX command line

- Linux/Unix is very stable - computers running Linux/Unix almost never crash
 - Linux/Unix is very efficient
 - it gets maximum number crunching power out of your processor (and multiple processors)
 - it can smoothly manage extremely huge amounts of data
 - it can give a new life to otherwise obsolete Macs and PCs
- Most new bioinformatics software is created for Unix - its easy for the programmers

Few free Bioinformatics SW for Linux



- **Linux** operating system, **mySQL** database
 - **Perl** - programming language
 - **Blast** and **Fasta** - similarity search
 - **Clustal** - multiple alignment
 - **Phylip** - phylogenetics
- **Phred/Phrap/Consed** - sequence assembly and SNP detection
- **EMBOSS** - a complete sequence analysis package created by the EMBL

Linux Basics



- ❑ Freely Downloadable from websites
- ❑ Available as sets of CDs
- ❑ Installation is very simple
- ❑ After installation you can create logins for different users
- ❑ Each user may login by his/her own login and passwd – own login area
- ❑ Upon login, default directory is home directory of the user

Linux basics..



- Linux/Unix is case sensitive i.e. WHO is not same as who
- Unix shell is a command program to communicate with a computer
- Shell interprets the command that you enter on keyboards
- Shell commands can be used to automate various programming tasks

Linux commands



- Usually short and cryptic like
 - vi or rm
- Commands may also have modifiers for advance options like:
 - "ls -l" and "mv -R" are different that "ls" or "mv" respectively

Wildcards

- You can substitute the `*` as a wildcard symbol for any number of characters in any filename.
- If you type just `*` after a command, it stands for all files in the current directory:

*lpr ** will print all files

- You can mix the `*` with other characters to form a search pattern:

ls a.txt* will list all files that start with “a” and end in “.txt”

- The “`?`” wildcard stands for any single character:

cp draft?.doc will copy *draft1.doc*, *draft2.doc*, *draftb.doc*, etc.

Control characters

- You type Control characters by holding down the 'control' key while also pressing the specified character.
 - While you are typing a command:
 - *ctrl-W* erases the previous word
 - *ctrl-U* erases the whole command line
- Control commands that work (almost) any time
 - *ctrl-S* suspends (halts) output scrolling up on your terminal screen
 - *ctrl-Q* resumes the display of output on your screen
 - *ctrl-C* will abort any program

Help on command line



- `man` : Type *man* and the name of a command to read the manual page for that command. e.g. “`man ls`”
- `apropos`: gives a list of commands that contain a given keyword in their man page header: e.g. “`apropos ls`”

Some important commands in Linux

- **ls**, Give a listing of the current directory. Try also `ls -l`
 - **cp**, Copy file from source to destination
- **mv**, Move file from source to destination. If both are the same directory, the file is renamed
- **vi**, Edit a file. `vi` is one of the most powerful text editors
 - **chmod**, Change file permissions
 - **mkdir**, **rmdir** Make/Remove a directory
 - **cd**, Change directory
 - **rm**, Remove a file. Can also remove directory tree
- **man ls**, Get help for `ls`. All commands have help

Networking



- **telnet**
 - Log into a remote host machine.
- **rlogin**
 - Almost the same as **telnet**, but uses a different protocol.
- **ping**
 - See if a remote host is up.
- **ftp**
 - Transfer files using the File Transfer Protocol.
- **netscape**
 - Run the Netscape web browser.
- **trn**
 - Read Internet News.
- **pine**
 - Read your mail using a full-screen display.
- **mail**
 - Read your mail using an ancient command-line program.
- **who**
 - See who else is logged in.
- **talk**
 - Talk to someone else who is current logged in.
- **lp**
 - Send a file or set of files to a printer.

Manipulating Files



- **cat**
 - Concatenate program. Can be used to concatenate multiple files together into a single file, or, much more frequently, to send the contents of a file to the terminal for viewing.
- **more**
 - Scroll through a file page by page. Very useful when viewing large files. Works even with files that are too big to be opened by a text editor.
- **less**
 - A version of **more** with more features.
- **head**
 - View the head (top) of a file. You can control how many lines to view.
- **tail**
 - View the tail (bottom) of a file. You can control how many lines to view. You can also use **tail** to view a growing file.
- **wc**
 - Count words, lines and/or characters in one or more files.
- **tr**
 - Substitute one character for another. Also useful for deleting characters.
- **sort**
 - Sort the lines in a file alphabetically or numerically.
- **uniq**
 - Remove duplicated lines in a file.
- **cut**
 - Remove sections from each line of a file or files.
- **fold**
 - Wrap each input line to fit in a specified width.
- **grep**
 - Filter a file for lines matching a specified pattern. Can also be reversed to print out lines that don't match the specified pattern.
- **gzip (gunzip)**
 - Compress (uncompress) a file.
- **tar**
 - Archive or unarchive an entire directory into a single file.
- **pico**
 - Run the pico text editor (good for beginners).
- **emacs**
 - Run the Emacs text editor (good for experts).

Text Editors Available on Linux Systems



□ **vi**

- Non-graphical (terminal-based) editor. Guaranteed to be available on any system. Requires knowledge of arcane keystroke commands. Distinctly unfriendly to novices.

□ **emacs**

- Window-based editor. Primitive menus make it slightly more friendly to novices. Still need to know keystroke commands to use. Installed on all Linux distributions and on most other Unix systems.

□ **xemacs**

- More sophisticated version of **emacs**, but usually not installed by default. All common commands are available from menus; however the user interface is still confusing at first. Very powerful editor, with built-in syntax checking, Web-browsing, news-reading, manual-page browsing, etc.

□ **pico**

- Simple terminal-based editor available on most versions of Unix. Uses keystroke commands, but they are listed in logical fashion at bottom of screen.

Computers in the facility



- Dual boot PCs
- Windows and Linux both
- Logins
 - Login: workshop
 - Passwd: whotdr05
- You may change your passwd using the command called "passwd"
- Start practicing !



Apache : Installation, Configuration, Basic Security

Click to add Text

Dr. M. Durairaj
Bharathidasan University



Outline

- Introduction
- Why Apache ?
- Installation
- Configuration
- Running Apache
- Basic Security



Introduction

- A basic web server works as follows :
 - It is a program that runs on a host computer .
 - It waits for a request from web browser/client for objects it has in its possession
 - Upon receiving the request (GET command from client), it retrieves the requested information and sends it to the client. The objects it can serve include HTML documents, plain text, images, sounds, video and other data.



Introduction

- Apache is the most popular web server on the internet, running approximately 60% of all web servers.
- This is the default web server on Red Hat, SuSE, and Debian systems and is well known in industry for its flexibility and performance.



Introduction

- Installing and maintaining is easy and ranks far below email and DNS in complexity & difficulty of administration.
- Its free and open source and full source code is available from Apache group site at www.apache.org.



Why Apache ?

- Apache's popularity is due to :
 - Apache is highly configurable .
 - It is extensible (for e.g. mod_perl and mod_php3 can be added) .
 - Supports virtual hosts or multi homed servers .
 - It is free and open source .



Installation

- Apache is included with most Linux distributions. On a machine installed with recent version of Linux, chances are Apache is already installed and running.
- Processor status can be checked to see if its running by using the command
machine1\$ **ps -ef | grep httpd**



Installation

- If one wishes to download the source code and compile it then,
 - Execute the **configure** script included with the distribution to detect type of system used and set up appropriate makefiles.
 - use **--prefix** option to specify where in ones directory tree the Apache server should live.



Installation

- For example :

```
% ./configure --prefix=/etc/httpd
```

- Default modules can be used or some features may be included or removed by invoking **-enable-module=** and **-disable-module=** options to `configure` .



Installation

- Some modules like asis, autoindex, env may be disabled for security reasons. A complete list of modules can be viewed at src/configuration file or

<http://www.apache.org/docs/mod/index.html>

- After executing **configure** run **make** and **make install** to actually compile and install the appropriate files .



Configuration

- All configuration files are in **conf** directory (/etc/httpd/conf). The files that are to be examined and customized are **httpd.conf**, **srm.conf** and **access.conf** .
- **httpd.conf** is used to set the TCP port (usually port 80), location of log files, and various network and performance parameters



Configuration

- **srm.conf** file defines the root of the directory tree in which servable documents are located .
- **access.conf** file manages security concerns. This file contains directives that control access on a per-file or per-directory basis .



Running Apache

- Apache can be started from machine's **rc** scripts or initiated by hand with

```
% /usr/sbin/httpd -f /etc/httpd/conf/httpd.conf
```

- It can be started automatically at boot time by making a link in **rc** directory that points to **/etc/init.d/httpd** .



Security

- Modifying the Default Header :
 - A hacker can exploit a web server by the information it sends in its header (version, machine type, its built-up etc) .
 - Its always better to modify the default header by changing the lines that reveals this information in **src/include/httpd.h** file .



Security

- Upgrading old software when necessary.
- Protecting Web Data with IP Restrictions :
 - Apache can be configured to allow restricted IP addresses only.
 - This can be done by adding following lines in .htaccess :

Order Deny, Allow

Deny from All

Allow from 192.168.1.100

Allow from 192.168.1.101



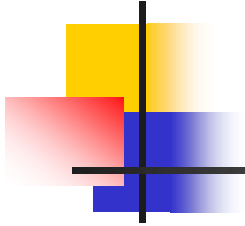
Security

- Using HTTP Authentication :
 - HTTP user authentication restricts access to a particular directory and subdirectories of the web server .
 - A browser implements authentication by prompting a dialog box for the user to type his username/password.



Security

- Using Secure HTTP Connections :
 - The Secure Socket Layer (SSL) should be used to minimize the likelihood that a hacker can snoop a username/password.
 - SSL not only encrypts the data before it is transferred to the web site, but also it decrypts the data received from the web site, thus securing the data transfers.



Questions ???

Unit 4: Manipulating MY SQL Database



DR. M. DURAIRAJ
BHARATHIDASAN UNIVERSITY

Introduction



- Many of the applications that a Web developer wants to use can be made easier by the use of a standardized database to store, organize, and access information.
- MySQL is an Open Source (GPL) Standard Query Language (SQL) database that is fast, reliable, easy to use, and suitable for applications of any size.
- SQL is the ANSI-standard database query language used by most databases (though all have their nonstandard extensions).
- MySQL can easily be integrated into Perl programs by using the Perl DBI (DataBase Independent interface) module.
- DBI is an Application Program Interface (API) that allows Perl to connect to and query a number of SQL databases (among them MySQL, mSQL, PostgreSQL, Oracle, Sybase, and Informix).

Tutorial



- Following the Swiss Army knife theory (20 percent of the functions give you 80 percent of the utility), a few SQL commands go a long way to facilitate learning MySQL/Perl/DBI.
- To illustrate these, we create a simple database containing information about some (fictional) people. Eventually, we'll show how to enter this information from a form on the Web, but for now we interface with SQL directly.
- First, try to make a connection to our MySQL server as the root MySQL user:
- **\$ mysql -u root**
- The MySQL root user is different from the Linux root user.
- The MySQL root user is used to administer the MySQL server only.
- If you see the following output:

```
ERROR 2002: Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)
```

- it likely means the MySQL server is not running.
- If your system is set up securely, it shouldn't be running, because you had no reason, before now, for it to be running.
- Use `chkconfig` as root to make sure it starts the next time the machine boots, and then start it by hand as follows:

```
# chkconfig mysqld on
```

```
# /etc/init.d/mysqld start
```

- Now you should be able to connect (*not* logged in as the Linux root user):

```
$ mysql -u root
```

- If not, see the MySQL log file at `/var/log/mysqld.log`.
- If so, you'll see a welcome message and the MySQL prompt:

Welcome to the MySQL monitor.

Commands end with `;` or `\g`.



• Your MySQL connection id is 3 to server version: 3.23.36
Type 'help;' or '\h' for help. Type '\c' to clear the buffer
mysql>

- As suggested, enter help; at the prompt.
- A list of MySQL commands (not to be confused with SQL commands) will be displayed.
- These allow you to work with the MySQL server.
- For grins, enter status; to see the status of the server.
- To illustrate these commands, we will create a database called people that contains information about people and their ages.



- First, we need to create the new database.
- Check the current databases to make sure a database of that name doesn't already exist; then create the new one, and verify the existence of the new database:

```
mysql> SHOW DATABASES;
```

```
+-----+  
| Database |  
+-----+  
| mysql |  
| test |  
+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> CREATE DATABASE people;
```

```
Query OK, 1 row affected (0.00 sec)
```



```
mysql> SHOW DATABASES;
```

```
+-----+  
| Database |  
+-----+  
| mysql |  
| people |  
| test |  
+-----+
```

```
3 rows in set (0.00 sec)
```

- SQL commands and subcommands (in the previous example, CREATE is a command; DATABASE is its subcommand) are case-insensitive.
- The name of the database (and table and field) are case sensitive.
- It's a matter of style whether one uses uppercase or lowercase, but traditionally the SQL commands are distinguished by uppercase.
- One way to think of a *database* is as a container for related *tables*.
- A table is a collection of *rows*, each row holding data for one *record*, each record containing chunks of information called *fields*.

The USE Command



- Before anything can be done with the newly created database, MySQL has to connect to it.
- That's done with the USE command:
- `mysql> USE people;`

The CREATE TABLE and SHOW TABLES Commands



- Each table within the database must be defined and created.
- This is done with the CREATE TABLE command.
- Create a table named age_information to contain an individual's first name, last name, and age.
- MySQL needs to know what kind of data can be stored in these fields.
- In this case, the first name and the last name are character strings of up to 20 characters each, and the age is an integer:

```
mysql> CREATE TABLE age_information (  
-> lastname CHAR(20),  
-> firstname CHAR(20),  
-> age INT  
-> );
```

Query OK, 0 rows affected (0.00 sec)

- It appears that the table was created properly (it says OK after all), but this can be checked by executing the SHOW TABLES command.
- If an error is made, the table can be removed with DROP TABLE.



- When a database in MySQL is created, a directory is created with the same name as the database (people, in this example):

```
# ls -l /var/lib/mysql
```

```
total 3
```

```
drwx----- 2 mysql mysql 1024 Dec 12 15:28 mysql
```

```
srwxrwxrwx 1 mysql mysql 0 Dec 13 07:19 mysql.sock
```

```
drwx----- 2 mysql mysql 1024 Dec 13 07:24 people
```

```
drwx----- 2 mysql mysql 1024 Dec 12 15:28 test
```



- Within that directory, each table is implemented with three files:

```
# ls -l /var/lib/mysql/people
```

```
total 10
```

```
-rw-rw---- 1 mysql mysql 8618 Dec 13 07:24 age_information.frm
```

```
-rw-rw---- 1 mysql mysql 0 Dec 13 07:24 age_information.MYD
```

```
-rw-rw---- 1 mysql mysql 1024 Dec 13 07:24 age_information.MYI
```

```
mysql> SHOW TABLES;
```

```
+-----+
```

```
| Tables_in_people |
```

```
+-----+
```

```
| age_information |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

- This example shows two MySQL datatypes: character strings and integers. Other MySQL data types include several types of integers

MySQL's data types



- TINYINT -128 to 127 (signed) or 0 to 255 (unsigned)
- SMALLINT -32768 to 32767 (signed) or 0 to 65535 (unsigned)
- MEDIUMINT -8388608 to 8388607 (signed) or 0 to 16777215 (unsigned)
- INTEGER (same as INT) -2147483648 to 2147483647 (signed) or 0 to 4294967295 (unsigned)
- BIGINT -9223372036854775808 to 9223372036854775807 (signed) or 0 to 18446744073709551615 (unsigned)

- **Floating points:**
 - FLOAT
 - DOUBLE
 - REAL (same as DOUBLE)
 - DECIMAL
 - NUMERIC (same as DECIMAL)

- **There are several data types to represent a date:**
 - DATE YYYY-MM-DD

- DATETIME YYYY-MM-DD HH:MM:SS
- TIMESTAMP YYYYMMDDHHMMSS or YYMMDDHHMMSS or YYYYMMDD or YYMMDD
- TIME HH:MM:SS
- YEAR YYYY or YY

- The table age_information used the CHAR character data type.
- The following are the other character data types.
- Several have LOB in their name— a BLOB is a Binary Large Object that can hold a variable amount of data.
- The types with TEXT in their name are just like their corresponding BLOBs except when matching is involved:
- The BLOBs are case-sensitive, and the TEXTs are case-insensitive.

- VARCHAR variable-length string up to 255 characters
- TINYBLOB maximum length 255 characters
- BLOB maximum length 65535 characters
- TINYTEXT
- TEXT
- MEDIUMBLOB maximum length 16777215 characters
- MEDIUMTEXT
- LONGBLOB maximum length 4294967295 characters
- LONGTEXT

The SELECT Command



- SELECT selects records from the database.
- When this command is executed from the command line, MySQL prints all the records that match the query.
- The simplest use of SELECT is shown in this example:

```
mysql> SELECT * FROM age_information
```

```
mysql> INSERT INTO age_information
```

```
-> (lastname, firstname, age)
```

```
-> VALUES ('Torvalds', 'Linus', 31);
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO age_information
```

```
-> (lastname, firstname, age)
```

```
-> VALUES ('Raymond', 'Eric', 40);
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM age_information;
```



- **mysql> SELECT * FROM age_information
-> ORDER BY lastname;**
- **mysql> SELECT lastname FROM age_information
-> ORDER BY lastname;**
- **mysql> SELECT age FROM age_information ORDER BY age DESC;**
- **mysql> SELECT lastname FROM age_information WHERE age > 35;**
- **mysql> SELECT lastname FROM age_information
-> WHERE age > 35 ORDER BY lastname;**

The UPDATE Command

- mysql> **SELECT * FROM age_information;**

```
+-----+-----+-----+
| lastname | firstname | age |
+-----+-----+-----+
| Wall | Larry | 46 |
| Torvalds | Linus | 31 |
| Raymond | Eric | 40 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

- mysql> **UPDATE age_information SET age = 47**
- -> **WHERE lastname = 'Wall';**
- Query OK, 1 row affected (0.00 sec)
- Rows matched: 1 Changed: 1 Warnings: 0
- mysql> **SELECT * FROM age_information;**

```
+-----+-----+-----+
| lastname | firstname | age |
+-----+-----+-----+
| Wall | Larry | 47 |
| Torvalds | Linus | 31 |
| Raymond | Eric | 40 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

The DELETE Command



- mysql> **DELETE FROM age_information WHERE lastname = 'Raymond';**

Query OK, 1 row affected (0.00 sec)

- mysql> **SELECT * FROM age_information;**

```
+-----+-----+-----+
| lastname | firstname | age |
+-----+-----+-----+
| Wall | Larry | 48 |
| Torvalds | Linus | 31 |
+-----+-----+-----+
```

2 rows in set (0.00 sec)

- Eric is in good company here, so put him back:

- mysql> **INSERT INTO age_information**

- > **(lastname, firstname, age)**

- > **VALUES ('Raymond', 'Eric', 40);**

Query OK, 1 row affected (0.00 sec)

- mysql> **SELECT * FROM age_information;**

```
+-----+-----+-----+
| lastname | firstname | age |
+-----+-----+-----+
| Wall | Larry | 48 |
| Torvalds | Linus | 31 |
| Raymond | Eric | 40 |
+-----+-----+-----+
```

3 rows in set (0.00 sec)

Dr. M. Durairaj
Associate Professor
School of Computer Science, Engineering and
Applications
Bharathidasan University

Databases, MySQL & PHP

Managing data

Building Data Dynamic Web Sites

- Truly dynamic web sites
 - Content changes over time
 - Content customised for individual user
 - Content automatically generated
- Content Programmatically generated
 - Can be File system based
 - HTML and Images stored on File System
 - Gets hard to manage over time
 - Database based
 - HTML, Images etc all generated from database
 - Easier to manage
 - If data is too large, can overload the database



Database?

Database

- **Structured collection of data.**
 - **Tables**
 - **Fields**
 - **Query**
 - **Reports**
- **Essentially a much more sophisticated implementation of the flat files.**

Relational Database

Relational Database

- Stores data in **separate tables** instead of a single store.
- **Relationships** between tables are set
- In theory, this provides a **faster, more flexible** database system.

Example

- We wish to maintain a **database** of student names, IDs, addresses, and any other information.
- Will be **updated frequently** with new names and information.
- Will want to **retrieve data** based on some predicate.
 - **e.g., ‘give me the names of all Massey students who live in Albany’.**
- Will want to update database with new information about students, not previously recorded.
 - **e.g., may decide we want to include IRD nos.**
- Very difficult to manage using ‘flat file’ systems

Databases

- **Fast, Efficient back end storage**
 - Easier to manage than file system based approach
- **Relational Database structure**
 - Well developed theory and practise
- **Multi-user capable**
 - Multithreaded, multiprocessor, sometimes cluster based systems
- **Standards based queries**
 - **Structured Query Language (SQL)**

MySQL Database

- world's most popular open source database because of its consistent **fast performance, high reliability and ease of use**
- Open Source License:- free
 - GNU General Public License
 - Free to modify and distribute but all modification must be available in source code format
- Commercial:- not free
 - Fully paid up professional support
- **used by Google, Facebook Nokia, YouTube, Yahoo!, Alcatel-Lucent, Zappos.com, etc.**

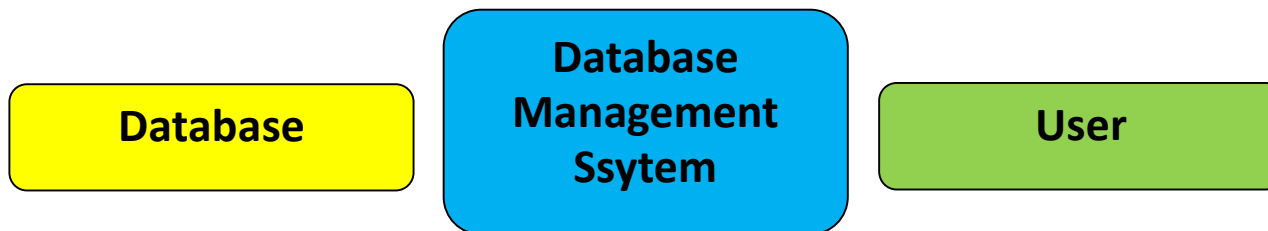
Basic Database Server Concepts

- **Database runs as a server**
 - Attaches to either a default port or an administrator specified port
- **Clients connect to database**
 - For secure systems
 - authenticated connections
 - usernames and passwords
- **Clients make queries on the database**
 - Retrieve content
 - Insert content
- **SQL (Structured Query Language)** is the language used to insert and retrieve content

Database Management System?

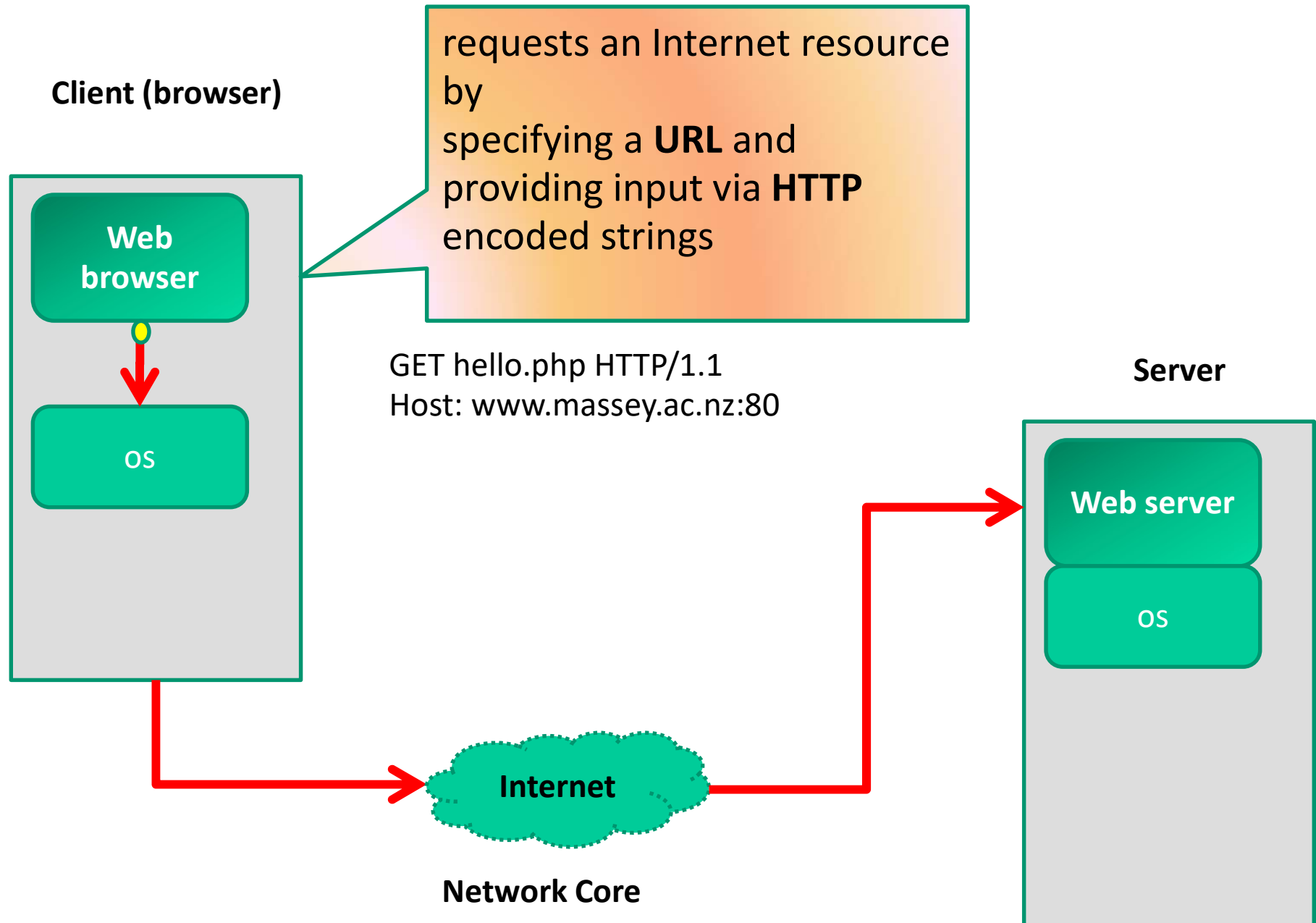
Database Management System

- Manages the storage and retrieval of data to and from the database and hides the complexity of what is actually going on from the user.



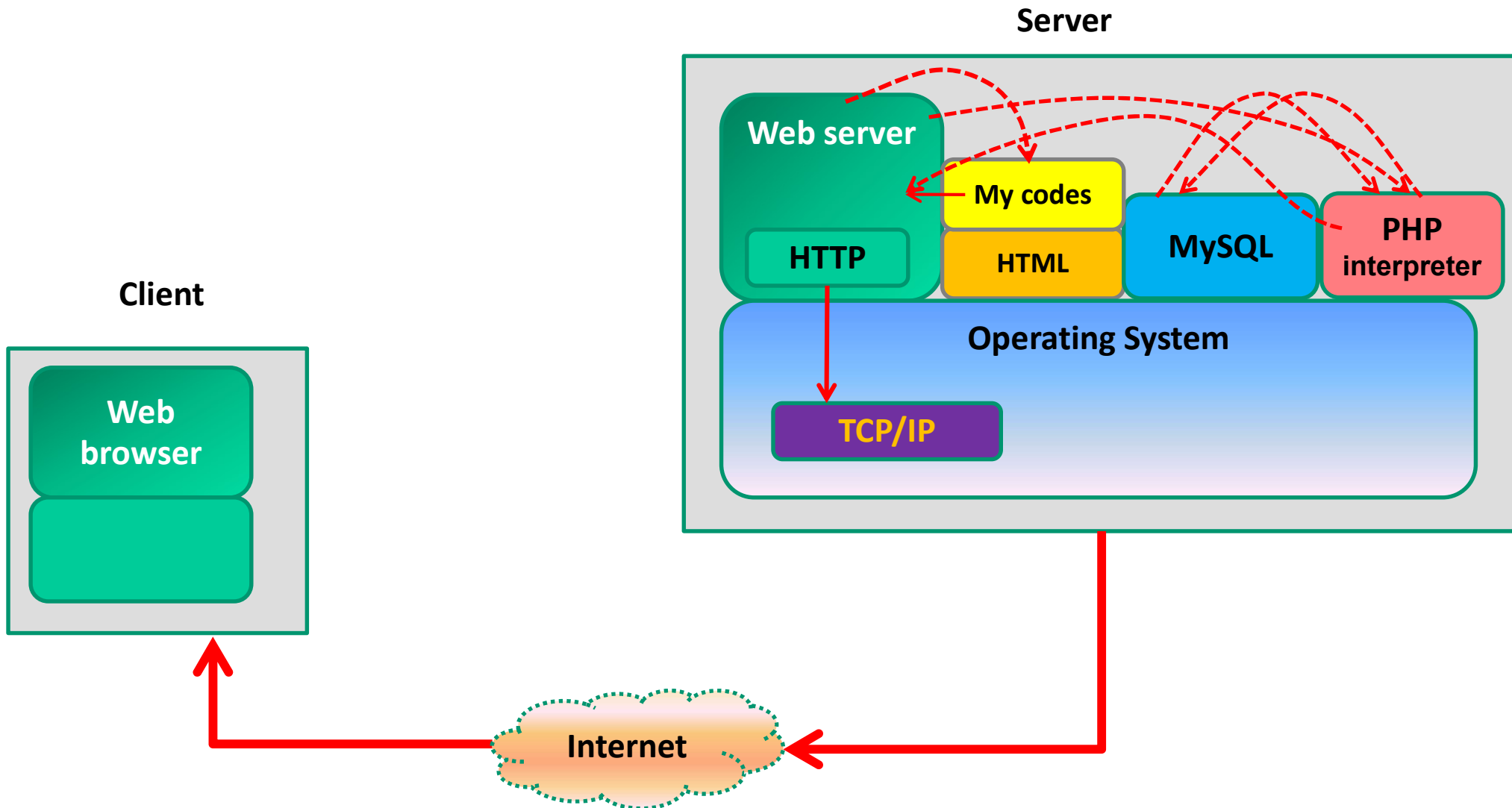
- **MySQL** is a relational database management system

Client: makes a request

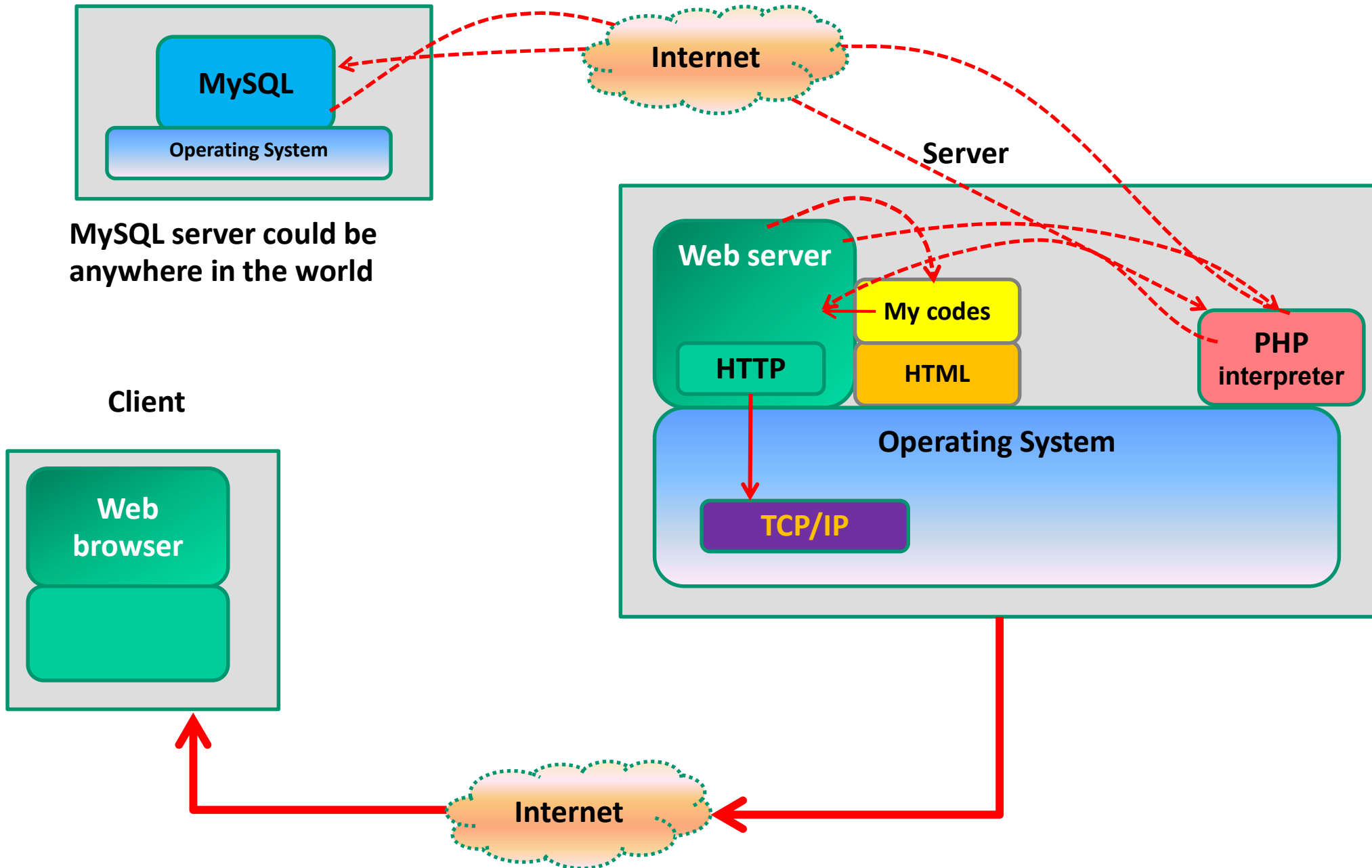


Server: responds

- Webserver supports HTTP.



Server: responds

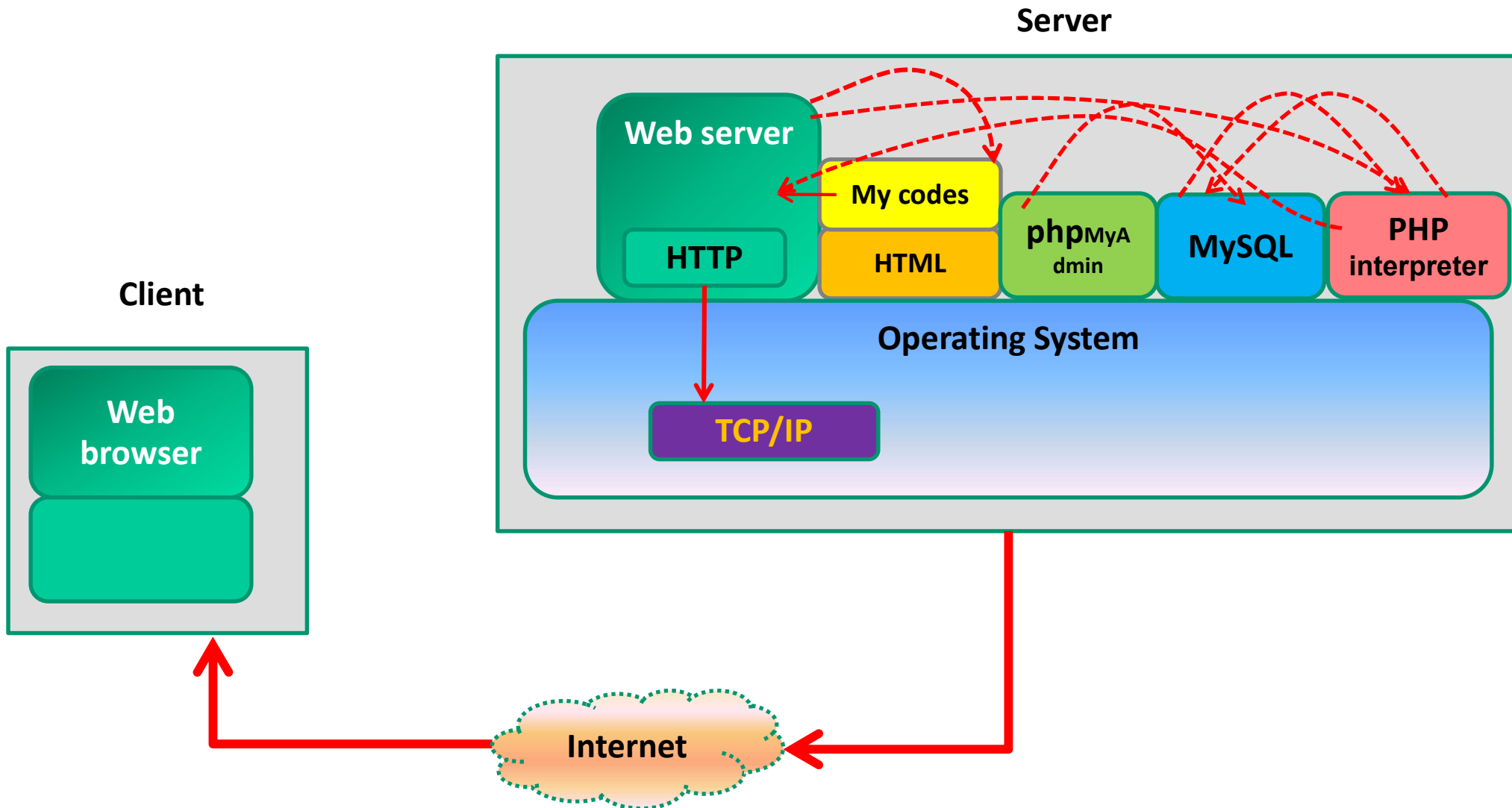


phpMyAdmin

- **MySQL can be controlled through a simple command-line interface; however, we can use phpMyAdmin as an interface to MySQL.**
- **phpMyAdmin is a very powerful tool; it provides a large number of facilities for customising a database management system.**

Server: responds

- Webserver supports HTTP.



Database Example

- **A Quick Tour**

Table: Customers (data)

			Id	Title	Surname	Firstname
<input type="checkbox"/>			1	Mrs	Smith	Lynne
<input type="checkbox"/>			4	Miss	Jones	Ann
<input type="checkbox"/>			5	Mr	Brown	Simon
<input type="checkbox"/>			6	Mr	Smith	David
<input type="checkbox"/>			7	Mr	Bell	Peter
<input type="checkbox"/>			8	Ms	Hall	Elizabeth
<input type="checkbox"/>			9	Mr	Smith	Kevin
<input type="checkbox"/>			10	Mr	Jones	Jack
<input type="checkbox"/>			11	Mr	Green	William
<input type="checkbox"/>			12	Mrs	Smith	Lynne
<input type="checkbox"/>			13	Mr	Bell	Simon
<input type="checkbox"/>			14	Mr	Brown	Ian
















Check All / Uncheck All *With selected:*

Table: Products (data)

	Id	Name	Description	Quantity	Cost
<input type="checkbox"/>  	1	Beer Glass	600 ml Beer Glass	345	3.99
<input type="checkbox"/>  	2	Wine Glass	125 ml Wine Glass	236	2.99
<input type="checkbox"/>  	3	Wine Glass	175 ml Wine Glass	436	3.5
<input type="checkbox"/>  	4	Shot Glass	50 ml Small Glass	132	1.5
<input type="checkbox"/>  	5	Spirit Glass	100 ml Short Glass	489	2.5
<input type="checkbox"/>  	6	Long Glass	200 ml Tall Glass	263	2.5
<input type="checkbox"/>  	7	Beer Glass	300 ml Beer Glass	247	2.99
<input type="checkbox"/>  	8	Wine Glass	225 ml Wine Glass	96	3.99

 [Check All](#) / [Uncheck All](#) *With selected:*   

Table: Purchases (data)

			Id	customers_Id	Day	Month	Year
<input type="checkbox"/>			1	2	3	9	2005
<input type="checkbox"/>			2	4	6	9	2005
<input type="checkbox"/>			3	6	13	9	2005
<input type="checkbox"/>			4	2	22	9	2005
<input type="checkbox"/>			5	1	28	9	2005
<input type="checkbox"/>			6	9	1	10	2005
<input type="checkbox"/>			7	7	1	10	2005





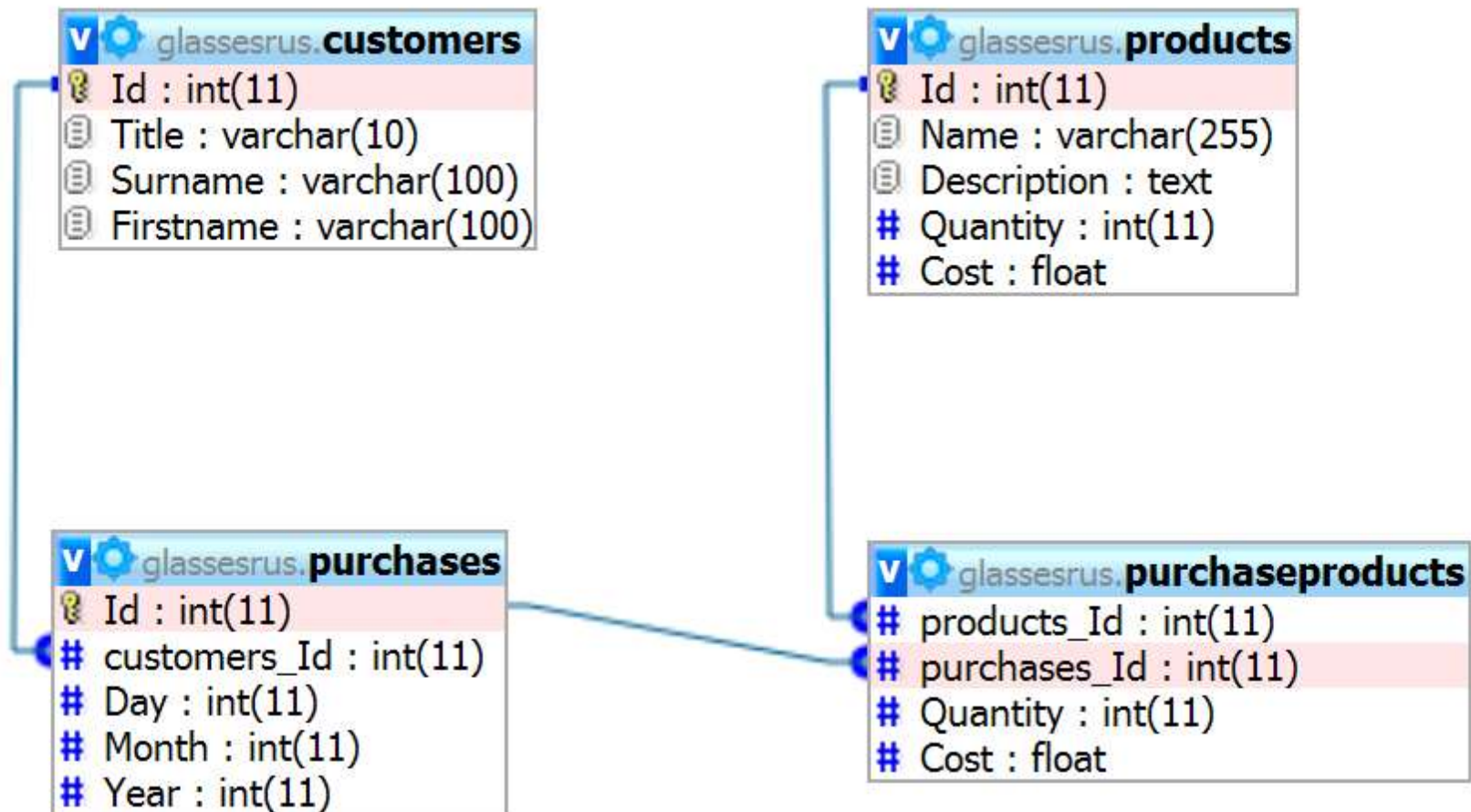
 [Check All / Uncheck All](#) *With selected:*   

Table: PurchaseProducts (data)

			products_id	purchases_id	Quantity	Cost
<input type="checkbox"/>			2	1	20	2.99
<input type="checkbox"/>			3	2	10	3
<input type="checkbox"/>			8	2	30	4.5
<input type="checkbox"/>			6	3	25	2.5
<input type="checkbox"/>			3	4	10	3.5
<input type="checkbox"/>			4	4	100	1.5
<input type="checkbox"/>			5	4	40	3
<input type="checkbox"/>			1	5	22	3.99
<input type="checkbox"/>			1	6	5	3.99
<input type="checkbox"/>			3	7	15	3.5
<input type="checkbox"/>			4	7	25	2
<input type="checkbox"/>			5	7	10	2.5
<input type="checkbox"/>			7	7	55	2.5
<input type="checkbox"/>			8	7	1	3.99

Check All / Uncheck All With selected:

Database Design



Database Field Types

In MySQL there are three main types :

- **text**
- **number**
- **Date/Time.**

Text Field Types

CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. Note: If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. Note: The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')

Numeric Field Types

TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

Date and Time Field Types

DATE()	A date. Format: YYYY-MM-DD Note: The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MM:SS Note: The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MM:SS Note: The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MM:SS Note: The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

phpMyAdmin

- **A Quick Tour**

phpMyAdmin

EASYPHP

PHP : MYSQL : APACHE FOR WINDOWS

english

5.3.3

www.easyphp.org

Support, download, faq, news, forum...

+ Support this project

PHP 5.3 migration guide

Most improvements in PHP 5.3.x have no impact on existing code. However, there are a few incompatibilities and new features that should be considered.

If you want to use EasyPHP on a USB key, you just need to copy the entire EasyPHP folder on the key. Be sure that all scripts are in the folder 'www' and your databases in 'mysql'.

PHP 5.3.3

APACHE 2.2.16

MYSQL 5.1.49

PHPMYADMIN 3.3.5

+ Manage MySQL

+ MySQL Parameters

+ PHP Parameters

+ Time Zone

+ Extensions

LOCAL WEB

C:\Program Files\EasyPHP-5.3.3\www\

Root

07273320

09213244

phptest

protected

protected2

temp

Create Database

The screenshot shows the phpMyAdmin interface for a MySQL server at 127.0.0.1. The left sidebar displays the phpMyAdmin logo, navigation icons (Home, SQL, Help, SQL), and the current database 'glassesrus (0)' with the message 'No tables found in database.' The main panel features a navigation bar with buttons for 'Databases', 'SQL', 'Status', 'Variables', 'Charsets', 'Eng', 'Processes', 'Export', 'Import', and 'Synchronize'. Below this is an 'Actions' section for 'MySQL 127.0.0.1'. The 'Create new database' form is visible, with the database name 'glassesRus' entered in the text field. A red arrow points to this text field. The 'Collation' dropdown is set to 'utf8_general_ci', and a 'Create' button is present to the right.

127.0.0.1

Databases SQL Status Variables Charsets Eng

Processes Export Import Synchronize

Actions

MySQL 127.0.0.1

Create new database ?

glassesRus Collation Create

MySQL connection collation: utf8_general_ci ?

Create Table: Customers



The screenshot shows the phpMyAdmin interface for a database named 'glassesrus'. The left sidebar displays the database name and a message: 'No tables found in database.' The main content area also shows 'No tables found in database.' Below this, there is a section titled 'Create new table on database glassesrus'. This section contains two input fields: 'Name: Customers' and 'Number of fields: 4'. A red arrow points to the 'Customers' text in the name field.

phpMyAdmin 127.0.0.1 ▶ glassesrus

Structure SQL Search Tracking Query Exp

~~Drop~~

No tables found in database.

No tables found in database.

Create new table on database glassesrus

Name: Customers Number of fields: 4

Specify the Table's Fields & Attributes: Customers

127.0.0.1 ▶ glassesrus ▶ Customers

Field	Type ?	Length/Values ¹	
Id	INT		No
Title	VARCHAR	10	No
Surname	VARCHAR	100	No
<u>Firstname</u>	VARCHAR	100	No

Table Edit Screen: Customers

127.0.0.1 ▶ glassesrus ▶ Customers

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Tracking](#) [Insert](#) [Export](#) [Import](#)

[Operations](#) [Empty](#) [Drop](#)

✓ Table `glassesrus`.`Customers` has been created.

```
CREATE TABLE `glassesrus`.`Customers` (  
  `Id` INT NOT NULL AUTO_INCREMENT ,  
  `Title` VARCHAR( 10 ) NOT NULL ,  
  `Surname` VARCHAR( 100 ) NOT NULL ,  
  `Firstname` VARCHAR( 100 ) NOT NULL ,  
  PRIMARY KEY ( `Id` )  
) ENGINE = MYISAM ;
```

[\[Edit \]](#) [\[Create PHP Code \]](#)

	Field	Type	Collation	Attributes	Null	Default	Extra		
<input type="checkbox"/>	Id	int(11)			No	None	AUTO_INCREMENT		
<input type="checkbox"/>	Title	varchar(10)	latin1_swedish_ci		No	None			
<input type="checkbox"/>	Surname	varchar(100)	latin1_swedish_ci		No	None			
<input type="checkbox"/>	Firstname	varchar(100)	latin1_swedish_ci		No	None			

↑ [Check All / Uncheck All](#) *With selected:*

Table: Products

127.0.0.1 ▶ glassesrus ▶ Products

Field	Type ?	Length/Values ¹	
Id	INT		↑
Name	VARCHAR	255	↑
Description	TEXT		↑
Quantity	INT		↑
Cost	FLOAT		↑

Table: Products

✓ Table `glassesrus`.`Products` has been created.

```
CREATE TABLE `glassesrus`.`Products` (  
  `Id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
  `Name` VARCHAR( 255 ) NOT NULL ,  
  `Description` TEXT NOT NULL ,  
  `Quantity` INT NOT NULL ,  
  `Cost` FLOAT NOT NULL  
) ENGINE = MYISAM ;
```

[[Edit](#)] [[Create PHP Code](#)]

	Field	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	Id	int(11)			No	<i>None</i>	AUTO_INCREMENT
<input type="checkbox"/>	Name	varchar(255)	latin1_swedish_ci		No	<i>None</i>	
<input type="checkbox"/>	Description	text	latin1_swedish_ci		No	<i>None</i>	
<input type="checkbox"/>	Quantity	int(11)			No	<i>None</i>	
<input type="checkbox"/>	Cost	float			No	<i>None</i>	

Insert Record: Customers

```
INSERT INTO `glassesrus`.`customers` (  
  `Id` ,  
  `Title` ,  
  `Surname` ,  
  `Firstname`  
)  
VALUES (  
  NULL , 'Mrs', 'Smith', 'Lynne'  
);
```

[Edit] [Create]


























Run SQL query/queries on database **glassesrus**: 





```
INSERT INTO `glassesrus`.`customers` (`Id`,  
`Title`, `Surname`, `Firstname`) VALUES (NULL,  
'Mrs', 'Smith', 'Lynne');
```

Fields

Id
Title
Surname
Firstname

Table: Customers (data)

			Id	Title	Surname	Firstname
<input type="checkbox"/>			1	Mrs	Smith	Lynne
<input type="checkbox"/>			4	Miss	Jones	Ann
<input type="checkbox"/>			5	Mr	Brown	Simon
<input type="checkbox"/>			6	Mr	Smith	David
<input type="checkbox"/>			7	Mr	Bell	Peter
<input type="checkbox"/>			8	Ms	Hall	Elizabeth
<input type="checkbox"/>			9	Mr	Smith	Kevin
<input type="checkbox"/>			10	Mr	Jones	Jack
<input type="checkbox"/>			11	Mr	Green	William
<input type="checkbox"/>			12	Mrs	Smith	Lynne
<input type="checkbox"/>			13	Mr	Bell	Simon
<input type="checkbox"/>			14	Mr	Brown	Ian

 **Check All / Uncheck All** *With selected:*   

Insert Record: Products

```
INSERT INTO `glassesrus`.`products` (  
  `Id` ,  
  `Name` ,  
  `Description` ,  
  `Quantity` ,  
  `Cost`  
)  
VALUES (  
  NULL , 'Beer Glass', '600 ml Beer Glass', '345', '3.99'  
);
```

[Edit] [Cre

Run SQL query/queries on database **glassesrus**: 

```
INSERT INTO `glassesrus`.`products` (`Id`, `Name`,  
`Description`, `Quantity`, `Cost`) VALUES (NULL, 'Beer  
Glass', '600 ml Beer Glass', '345', '3.99');
```

Fields


Id
Name
Description
Quantity
Cost

Table: Products (data)

	Id	Name	Description	Quantity	Cost
<input type="checkbox"/>  	1	Beer Glass	600 ml Beer Glass	345	3.99
<input type="checkbox"/>  	2	Wine Glass	125 ml Wine Glass	236	2.99
<input type="checkbox"/>  	3	Wine Glass	175 ml Wine Glass	436	3.5
<input type="checkbox"/>  	4	Shot Glass	50 ml Small Glass	132	1.5
<input type="checkbox"/>  	5	Spirit Glass	100 ml Short Glass	489	2.5
<input type="checkbox"/>  	6	Long Glass	200 ml Tall Glass	263	2.5
<input type="checkbox"/>  	7	Beer Glass	300 ml Beer Glass	247	2.99
<input type="checkbox"/>  	8	Wine Glass	225 ml Wine Glass	96	3.99


 [Check All](#) / [Uncheck All](#) *With selected:*   

Edit Record

 0 row(s) affected.

```
UPDATE `glassesrus`.`customers` SET `Firstname` = 'Elizabeth' WHERE `customers`.`Id` =8;
```

[\[Edit \]](#) [\[Crea](#)

 Showing rows 0 - 11 (12 total, Query took 0.0008 sec)

```
SELECT *  
FROM `customers`  
LIMIT 0 , 30
```


Export

Export

Select All / Unselect All

customers
products

CodeGen
 CSV
 CSV for MS Excel
 Microsoft Word 2000
 LaTeX
 MediaWiki Table
 Open Document Spreadsheet
 Open Document Text
 PDF
 PHP array
 SQL
 Taxy! text
 Excel 97-2003 XLS Workbook
 Excel 2007 XLSX Workbook
 XML
 YAML

Options

Add custom comment into header (\n splits lines)

Comments
 Enclose export in a transaction
 Disable foreign key checks
SQL compatibility mode

Structure

Add DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT
 Add IF NOT EXISTS
 Add AUTO_INCREMENT value
 Enclose table and field names with backquotes
 Add CREATE PROCEDURE / FUNCTION / EVENT

Add into comments

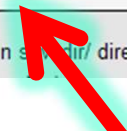
Creation/Update/Check dates
 Relations
 MIME type

Data

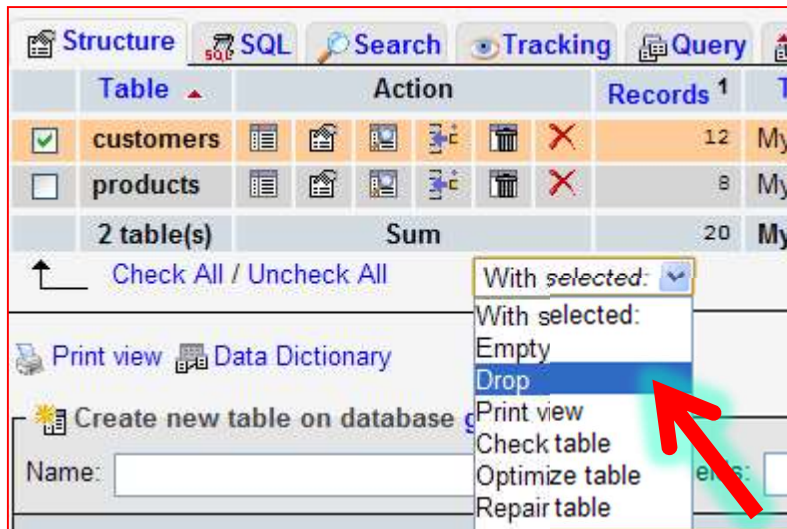
Complete inserts
 Extended inserts
Maximal length of created query

Use delayed inserts
 Use ignore inserts
 Use hexadecimal for BLOB
Export type

Save as file
 Save on server in s...dir/ directory



Deleting a Table



Restoring a database from an SQL file

File to import

Location of the text file No file chosen (Max:)

Character set of the file:

Imported file compression will be automatically detected from: None, gzip, b

Partial import

Allow the interruption of an import in case the script detects it is close to t however it can break transactions.

Number of records (queries) to skip from start

Format of imported file

CSV

Open Document Spreadsheet

SQL

Excel 97-2003 XLS Workbook

Excel 2007 XLSX Workbook

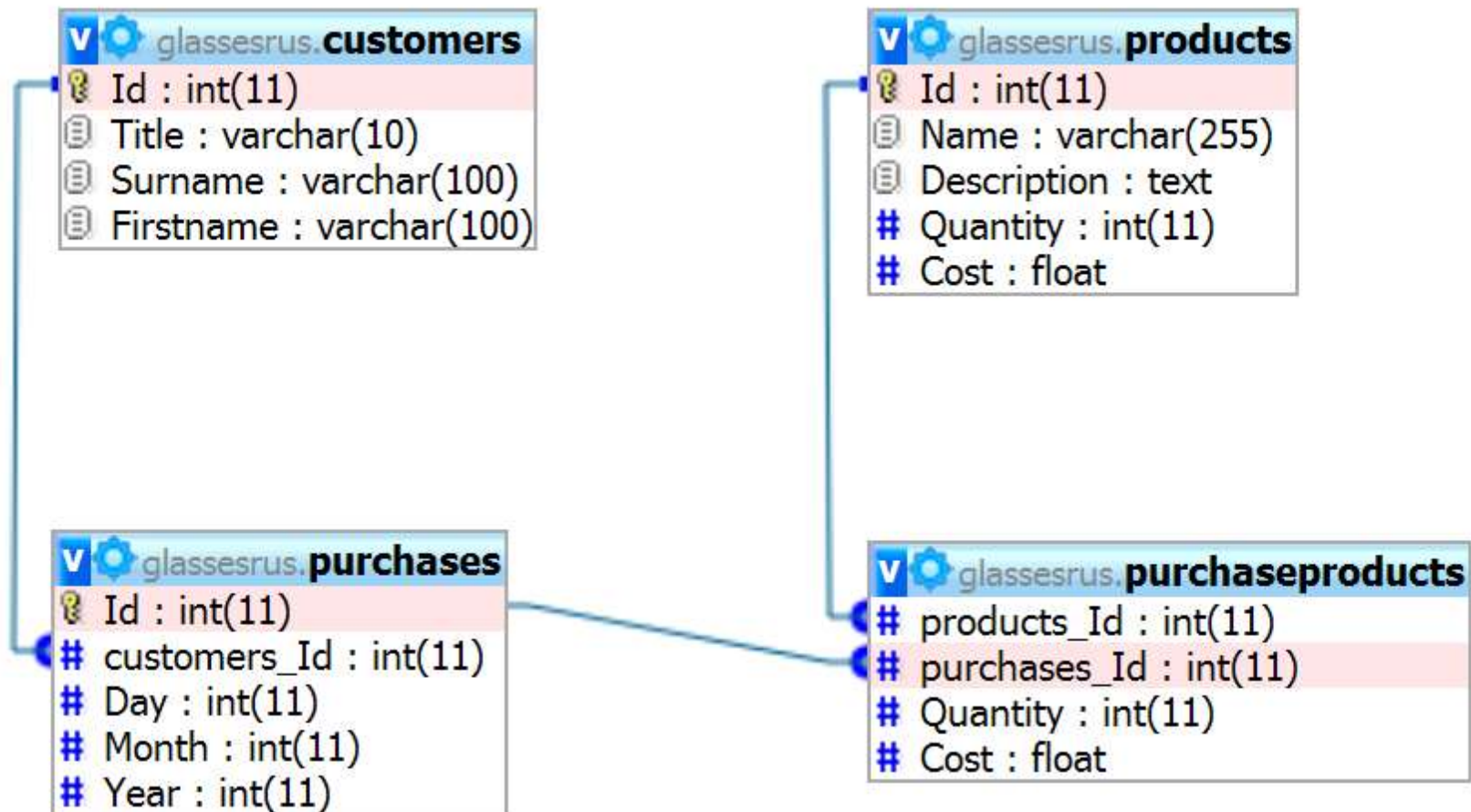
XML

Options

SQL compatibility mode

Do not use AUTO_INCREMENT fo

Database Design



Summary

- Concept of databases
- Tables and Fields
- Field Types
- phpMyAdmin Tool for manipulating databases
- Creation of a database
- How to add and edit records
- How to back-up a database
- Database Design

MySQL and PHP

Connecting to a MySQL DBMS

- **In order for our PHP script to access a database we need to form a connection from the script to the database management system.**

```
resourceId = mysql_connect(server, username, password);
```

- Server is the DBMS server
- username is your username
- password is your password

Connecting to a MySQL DBMS

- In order for our PHP script to access a database we need to form a connection from the script to the database management system.

```
resourceId = mysql_connect(server, username, password);
```

- The function returns a resource-identifier type.
- a PHP script can connect to a **DBMS** anywhere in the world, so long as it is connected to the internet.
- we can also connect to multiple DBMS at the same time.

Selecting a database

- **Once connected to a DBMS, we can select a database.**

```
mysql_select_db(databasename, resourceId);
```

- the resourceId is the one returned by mysql_connect()
- the function returns true if the selection succeeded; false, otherwise.

Example: Connect to a DBMS and access database

```
<?php
$dbLocalhost = mysql_connect("localhost", "root", "")
    or die("Could not connect: " . mysql_error());
mysql_select_db("glassesrus", $dbLocalhost)
    or die("Could not find database: " . mysql_error());
echo "<h1>Connected To Database</h1>";
?>
```

- **die()** stops execution of script if the database connection attempt failed.
- **mysql_error()** returns an error message from the previous MySQL operation.

Reading from a database

- **We can now send an SQL query to the database to retrieve some data records.**

```
resourceRecords = mysql_query(query, resourceId);
```

- the resourceId is the one returned by mysql_connect()
- the function returns a resource identifier to the returned data.

Example: Connect to a DBMS, access database, send query

```
<?php
$dbLocalhost = mysql_connect("localhost", "root", "")
    or die("Could not connect: " . mysql_error());
mysql_select_db("glassesrus", $dbLocalhost)
    or die("Could not find database: " . mysql_error());
$dbRecords = mysql_query("SELECT * FROM customers", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());
echo "<h1>Connected To Database</h1>";
?>
```

- **the function will return a resource pointer (not the actual data) to all the records that match the query.**
- **If all goes well, this script will output nothing on screen.**

Extract contents of one record

- **We can now extract the actual data from the resource pointer returned by `mysql_query()`.**

```
fieldData= mysql_result(resourceRecords, row, field);
```

- the `resourceRecords` is the one returned by `mysql_query()`
- `field` – database field to return
- the function returns the **data stored in the field**.

Example: Connect to a DBMS, access database, send query

```
<?php
$dbLocalhost = mysql_connect("localhost", "root", "")
    or die("Could not connect: " . mysql_error());
mysql_select_db("glassesrus", $dbLocalhost)
    or die("Could not find database: " . mysql_error());
$dbRecords = mysql_query("SELECT * FROM customers", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());
$strSurname = mysql_result($dbRecords, 0, "Surname");
echo "<p>$strSurname</p>";
?>
```

- the function will return a resource pointer (not the actual data) to all the records that match the query.
- If all goes well, this script will output a surname on screen.

SQL statement

```
SELECT * FROM customers
```

- Go and **obtain** from the database
- **every** field
- **FROM** the
- **customers** table

Separating the database connection

It is worth separating the database connectivity from our scripts and placing it in a separate file.

- It provides a convenient means of moving your scripts from one database platform to another.

Example: Separating the database connection

```
<?php
// File: database2.php
$strLocation = "Home";
//$strLocation = "Work";
if ($strLocation == "Home") {
    $dbLocalhost = mysql_connect("localhost", "root", "")
        or die("Could not connect: " . mysql_error());
    mysql_select_db("glassesrus", $dbLocalhost)
        or die("Could not find database: " . mysql_error());
} else {
    $dbLocalhost = mysql_connect("localhost", "username", "password")
        or die("Could not connect: " . mysql_error());

    mysql_select_db("anotherdatabase", $dbLocalhost)
        or die("Could not find database: " . mysql_error());
}
?>
```

- **\$strLocation** could be easily switched between 'Home' or 'Work'

Viewing a whole record

To view the whole record returned from `mysql_query()`, we need another function...

```
array = mysql_fetch_row(resourceRecords)
```

- `resourceRecords` – resource identifier returned from `mysql_query()`.
- it returns an array containing the database record.

Example: Displaying all customer records

```
<?php

require_once("database2.php");

$dbRecords = mysql_query("SELECT * FROM customers", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());

while ($arrRecord = mysql_fetch_row($dbRecords)) {
    echo "<p>" . $arrRecord[0] . " ";
    echo      $arrRecord[1] . " ";
    echo      $arrRecord[2] . " ";
    echo      $arrRecord[3] . "</p>";
}
?>
```

- The function returns false when the last record is returned; thus, stopping the loop.
- Note, however, that the fields are referred to by using **numbers** – not very easy to read and mistakes can be introduced.

Limiting the records returned

```
SELECT Surname FROM customers
```

- Retrieves only the Surname field from the table customers

Limiting the records returned

```
SELECT * FROM customers LIMIT 3,4
```

- Select a certain number of records from a table
- 3 is the starting row
- 4 is the number of records to be selected after the starting row

Searching for matching records

```
SELECT * FROM customers WHERE Title='Mr'
```

- The WHERE attribute specifies what to search for within the database records.
- in this example, only records which have a title of 'Mr' will be returned.

Searching for matching records

```
SELECT * FROM customers WHERE Title='Mr' OR  
Title='Mrs'
```

- The WHERE attribute specifies what to search for within the database records.
- in this example, only records which have a title of 'Mr' or 'Mrs' will be returned.
- we can also use **AND** and **OR** to formulate more sophisticated conditions.
-

Searching for matching records

```
SELECT * FROM customers WHERE Title='Mr' AND  
Surname='Smith' OR Title='Mrs'
```

- The WHERE attribute specifies what to search for within the database records.
- in this example, only records which have a surname of 'Smith' and title of 'Mr' or the title of 'Mrs' will be returned.
- we can also use **AND** and **OR** to formulate more sophisticated conditions.
-

Sorting records

The **ORDER BY** attribute can be used to sort the order in which records are obtained.

```
SELECT * FROM cutomers ORDER BY Surname DESC
```

- the **ORDER BY** attribute is followed by the data field on which to sort the record
- **DESC** or **ASC** – from high to low, or from low to high

Accessing Multiple Tables

```
<?php
// File: example15-13.php

require_once("database2.php");

$dbRecords = mysql_query("SELECT * FROM customers WHERE Title = 'Mrs'", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());

echo "<p>Customers:</p>";
while ($arrRecords = mysql_fetch_array($dbRecords)) {
    echo "<p>" . $arrRecords["Id"] . " ";
    echo $arrRecords["Title"] . " ";
    echo $arrRecords["Surname"] . " ";
    echo $arrRecords["Firstname"] . "</p>";
}

//...continued...
```

Accessing Multiple Tables

```
//continuation...

$dbRecords = mysql_query("SELECT * FROM products WHERE Name = 'Wine Glass'",
$dbLocalhost)
    or die("Problem reading table: " . mysql_error());

echo "<p>Products:</p>";
while ($arrRecords = mysql_fetch_array($dbRecords)) {
    echo "<p>" . $arrRecords["Id"] . " ";
    echo $arrRecords["Name"] . " ";
    echo $arrRecords["Description"] . " ";
    echo $arrRecords["Quantity"] . " ";
    echo $arrRecords["Cost"] . "</p>";
}
?>
```

Using records to read another table

Read a customer record, and then show the products purchased by that customer.

Tables

- Customers
- Products
- Purchases
- PurchaseProducts

Using records to read another table

```
...
$strSurname = "Jones";
$dbCustRecords = mysql_query("SELECT * FROM customers WHERE Surname = '$strSurname' ",...)
while ($arrCustRecords = mysql_fetch_array($dbCustRecords)) { ##1
    $intId = $arrCustRecords["Id"];
    //display customer's details
    $dbPurRecords = mysql_query("SELECT * FROM purchases WHERE customers_Id = '$intId'", ...)
while ($arrPurRecords = mysql_fetch_array($dbPurRecords)) { ##2
    $intPurId = $arrPurRecords["Id"];
    //display purchase date
    $dbProRecords=mysql_query("SELECT * FROM purchaseProducts WHERE purchases_Id='$intPurId' ",...)
while ($arrProRecords = mysql_fetch_array($dbProRecords)) { ##3
    $intProductId = $arrProRecords["products_Id"];
    //display Quantity
    $dbProductRecords = mysql_query("SELECT * FROM products WHERE Id = '$intProductId'",...)
    $arrProductRecord = mysql_fetch_array($dbProductRecords);
    //display product details
} #3
} #2
} ##1
```

Using records to read another table

```
<?php
require_once("database2.php");

$strSurname = "Jones";

$dbCustRecords = mysql_query("SELECT * FROM customers WHERE Surname = '$strSurname'
", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());

while ($arrCustRecords = mysql_fetch_array($dbCustRecords)) {
    $intId = $arrCustRecords["Id"];
    echo "<p>Customer: ";
    echo $arrCustRecords["Title"] . " ";
    echo $arrCustRecords["Surname"] . " ";
    echo $arrCustRecords["Firstname"] . "</p>";

    $dbPurRecords = mysql_query("SELECT * FROM purchases WHERE customers_Id = '$intId'",
$dbLocalhost)
        or die("Problem reading table: " . mysql_error());
```

Using records to read another table

```

while ($arrPurRecords = mysql_fetch_array($dbPurRecords)) {
    $intPurId = $arrPurRecords["Id"];
    echo "<p>Purchased On: ";
    echo $arrPurRecords["Day"] . "/";
    echo $arrPurRecords["Month"] . "/";
    echo $arrPurRecords["Year"] . "</p>";

    $dbProRecords= mysql_query("SELECT * FROM purchaseProducts WHERE purchases_Id='$intPurId' ",
$dblocalhost)
    or die("Problem reading table: " . mysql_error());

    while ($arrProRecords = mysql_fetch_array($dbProRecords)) {
        $intProductId = $arrProRecords["products_Id"];
        echo "<p>" . $arrProRecords["Quantity"] . " ";

        $dbProductRecords = mysql_query("SELECT * FROM products WHERE Id = '$intProductId'",
$dblocalhost)
        or die("Problem reading table: " . mysql_error());

        $arrProductRecord = mysql_fetch_array($dbProductRecords);
        echo $arrProductRecord["Name"] . " (" . $arrProductRecord["Description"] . ") at &#163;";
        echo $arrProRecords["Cost"] . " each.</p>";
    }
}
?>

```

Inserting records

How to create new database records and insert them into a table?

INSERT INTO table (field1, field2,...) VALUES ('value1', 'value2',...)

- Alternatively, we have a simplified syntax:

INSERT INTO table VALUES ('value1', 'value2',...)

```
$dbProdRecords = mysql_query("INSERT INTO products  
VALUES ( ' ', 'Beer Mug', '600 ml Beer Mug', '100', '5.99')",  
$dbLocalhost)
```


Inserting records

```
<?php
// File: example15-15.php

require_once("database2.php");

$dbProdRecords = mysql_query("INSERT INTO products VALUES ('', 'Beer Mug', '600
ml Beer Mug', '100', '5.99')", $dbLocalhost)
    or die("Problem writing to table: " . mysql_error());

$dbProdRecords = mysql_query("SELECT * FROM products", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());

while ($arrProdRecords = mysql_fetch_array($dbProdRecords)) {
    echo "<p>" . $arrProdRecords["Id"] . " ";
    echo $arrProdRecords["Name"] . " ";
    echo $arrProdRecords["Description"] . " ";
    echo $arrProdRecords["Quantity"] . " ";
    echo $arrProdRecords["Cost"] . "</p>";
}
?>
```

Deleting records

How to delete database records from tables?

DELETE FROM table WHERE field='value'

e.g.

```
$dbCustRecords = mysql_query("DELETE FROM customers  
WHERE Id='3'", $dbLocalhost)
```

Note: If you have a relational database, you should tidy-up the other tables, based on their connection with the record you've deleted.

Deleting records

How to delete database records from tables?

DELETE FROM table

This will delete all records from a table!

Note: back-up your database first!

Amending records

How to modify the contents of an existing database record?

UPDATE table **SET** field='value1', field='value2'...**WHERE**
field='value'

- requires you to specify the table, the list of fields with their updated values, and a condition for selection (WHERE).

Amending records

```
<?php
// File: example15-18.php

require_once("database2.php");

$dbCustRecords = mysql_query("UPDATE products SET Description='250 ml Tall
Glass' WHERE Id='6'", $dbLocalhost)
    or die("Problem updating table: " . mysql_error());

$dbProdRecords = mysql_query("SELECT * FROM products", $dbLocalhost)
    or die("Problem reading table: " . mysql_error());

while ($arrProdRecords = mysql_fetch_array($dbProdRecords)) {
    echo "<p>" . $arrProdRecords["Id"] . " ";
    echo $arrProdRecords["Name"] . " ";
    echo $arrProdRecords["Description"] . " ";
    echo $arrProdRecords["Quantity"] . " ";
    echo $arrProdRecords["Cost"] . "</p>";
}
?>
```

Amending records

How to modify the contents of an existing database record?

```
UPDATE table SET field='value1', field='value2'...WHERE  
field='value'
```

Another Example:

```
$dbCustRecords = mysql_query("UPDATE products SET Name='Beer  
and Lager Glass' WHERE Name='Beer Glass'", $dbLocalhost)
```

- A number of records will be updated in this example.

Counting the number of records

How to count the number of records after running a query?

```
$dbProdRecords = mysql_query("SELECT * FROM products",  
$dbLocalhost)
```

```
or die("Problem reading table: " . mysql_error());
```

```
$intProductCount = mysql_num_rows($dbProdRecords);
```

- you can also use the same function to determine if a record exists.

Example15-20.php

Example15-21.php

Select a substring

How to count the number of records after running a query?

```
SELECT * FROM products WHERE substring(Name,1,4)='Wine'
```

- This will return all records from the products table where the first four characters in the name field equals 'Wine'

End of Lecture



Unit V: Working with PHP

PHP



1-1

Dr. M. Durairaj
Bharathidasan University

PHP

- PHP overview
- PHP General Syntactic Characteristics
- PHP output to browser
- Primitives, Operations, and Expressions
- Control Statement
- Array
- Function
- File access
- Cookie
- Session
- Form process

ORIGINS AND USE OF PHP

○ *Origins*

- Rasmus Lerdorf – 1994
 - Developed to allow him to track visitors to his Web site
- An open-source product
- An acronym for *Personal Home Page*, or *PHP: Hypertext Preprocessor*
- PHP is a server-side scripting language whose scripts are embedded in HTML documents
 - Similar to JavaScript, but on the server side
- Used for form handling, file processing, and database access

ORIGINS AND USE OF PHP

- PHP is “A” server-side scripting language
 - One of an alternative to CGI, ASP.NET (Active server pages), and JSP (Java Server Pages)
- The PHP processor has two modes:
 - copy (XHTML) and interpret (PHP)
- PHP syntax is similar to that of JavaScript
- PHP is dynamically typed
- PHP is a interpreted language
 - Programs may be executed from source form
 - Each instruction is immediately translated and acted upon by the computer

GENERAL SYNTACTIC CHARACTERISTICS

- PHP code can be specified in an XHTML document internally or externally: myphp.php
 - Internally: `<?php ... ?>`
 - Can appear almost everywhere
 - Externally: `include ("myScript.inc")`
 - The included file can have both PHP and XHTML, if the file has PHP, the PHP must be in `<?php .. ?>`, even if the include is already in `<?php .. ?>`
 - Variable conflict
- PHP mode of operation
 - Copy mode
 - Interpret mode
- Every variable name begin with a \$
 - Case sensitive
 - A letter or an underscore followed by any number of letters, digits, or underscores.

GENERAL SYNTACTIC CHARACTERISTICS

- *Comments* - three different kinds (Java and Perl)

// ...

...

/* ... */

- Statements are terminated with semicolons
- Compound statements are formed with braces
 - Unless used as the body of a function definition, compound statements cannot be blocks (cannot define locally scoped variables)

OUTPUT

- Output from a PHP script is HTML that is sent to the browser
 - HTML is sent to the browser through standard output
 - There are three ways to produce output: echo, print, and printf
- echo and print take a string, but will coerce other values to strings
- ```
$name="John"; $age=20;
```
- echo "\$name", "\$age"; (any number of parameters)
  - echo("my name: \$name, my age: \$age"); (only one)
  - print "\$name and \$age";
  - print ("my name: \$name, my age: \$age");
  - printf("my name: %s, my age: %s", \$name, \$age);
- Echo does not return a value; print return 1 or 0; printf returns the length of the outputted string

PHP

[Output.php](#)



# VAR\_DUMP

- Dumps information about a variable
  - `$var1=3.1;`
    - `Var_dump($var1);`
    - `Float(3.1);`
  - `$var2="3.1";`
    - `Var_dump($var2);`
    - `String("3.2");`

# PHP

- PHP overview
- PHP General Syntactic Characteristics
- PHP Output to browser
- **Primitives, Operations, and Expressions**
- Control Statement
- Array
- Function
- File access
- Cookie
- Session
- Form process

# PRIMITIVES, OPERATIONS, AND EXPRESSIONS

- *Variables* [primitive.php](#)
  - No type declarations
  - An unassigned (unbound) variable has the value: NULL
  - The unset function sets a variable to NULL
  - The IsSet function is used to determine whether a variable is NULL
  - `error_reporting(15);` - prevents PHP from using unbound variables
- PHP has many predefined variables, including the environment variables of the host operating system
  - You can get a list of the predefined variables by calling `phpinfo()` in a script

# PRIMITIVES, OPERATIONS, AND EXPRESSIONS

- *There are eight primitive types:*
  - Four scalar types: Boolean, integer, double, and string
  - Two compound types: array and object
  - Two special types: resource and NULL

| Data type     | Description                                                      |
|---------------|------------------------------------------------------------------|
| int, integer  | Whole numbers (i.e., numbers without a decimal point).           |
| float, double | Real numbers (i.e., numbers containing a decimal point).         |
| string        | Text enclosed in either single ( ' ' ) or double ( " " ) quotes. |
| bool, Boolean | True or false.                                                   |
| array         | Group of elements of the same type.                              |
| object        | Group of associated data and methods.                            |
| Resource      | An external data source.                                         |
| NULL          | No value.                                                        |

Fig. 26.2 PHP data types.

# PRIMITIVES, OPERATIONS, AND EXPRESSIONS

## ○ *Strings*

[string.php](#)

- Characters are single bytes
- The length of a string is limited only by the available memory
- String literals use single or double quotes
  - *Single-quoted string literals*
    - Embedded variables are NOT interpolated
    - Embedded escape sequences are NOT recognized
  - *Double-quoted string literals*
    - Embedded variables ARE interpolated
    - If there is a variable name in a double quoted string but you don't want it interpolated, it must be backslashed
    - Embedded escape sequences ARE recognized
- For both single- and double-quoted literal strings, embedded delimiters must be backslashed
- String character access
  - `$str="Apple"`
  - `$str{2}="p"`

PHP

# PRIMITIVES, OPERATIONS, AND EXPRESSIONS

- Boolean [boolean.php](#)
  - values are true and false (case insensitive)
  - 0 and "" and "0" are false; others are true
    - But "0.0" is true
- Arithmetic Operators and Expressions
  - Usual operators
  - If the result of integer division is not an integer, a double is returned
  - Any integer operation that results in overflow produces a double
  - The modulus operator coerces its operands to integer, if necessary
- Arithmetic functions
  - floor, ceil, round, abs, min, max, rand, etc.
    - Round(\$val, x);

# PRIMITIVES, OPERATIONS, AND EXPRESSIONS

## ○ *Scalar Type Conversions*

[conversion.php](#)

- *String to numeric*
  - If the string contains an e or an E, it is converted to double; otherwise to integer
  - If the string does not begin with a sign or a digit, zero is used
- **Explicit conversions – casts**
  - e.g., (int)\$total or intval(\$total) or settype(\$total, "integer")
    - intval(\$total), doubleval(\$total), strval(\$total);
- **The type of a variable can be determined with gettype or is\_type**
  - gettype(\$total) - it may return "unknown"
  - is\_integer(\$total) – a predicate function
    - is\_double(), is\_bool(), is\_string()

# PHP

- PHP overview
- PHP General Syntactic Characteristics
- PHP Output to browser
- Primitives, Operations, and Expressions
- **Control Statement**
- Array
- Function
- File access
- Cookie
- Session
- Form process



# CONTROL STATEMENT

- Control Expressions
  - Relational operators - same as JavaScript
    - >, <, >=, <=, !=, ==
  - Boolean operators
    - And, or, xor, !, &&, and ||
- Selection statements
  - if, elseif, else
  - switch - as in C
    - The switch expression type must be integer, double, or string
- Loop statements
  - while - just like C
  - do-while - just like C
  - for - just like C
  - foreach - discussed later

# CONTROL STATEMENT

- break
  - in any for, foreach, while, do-while, or switch
- continue
  - in any loop
- Alternative compound delimiters – more readability
  - if(...):
  - ...
  - endif;

[Powers.php](#)

# INTERMINGLE

- XHTML can be intermingled with PHP

```
<?php Intermingle.php
```

```
 $a = 7;
```

```
 $b = 7;
```

```
 if ($a == $b)
```

```
 {
```

```
 $a = 3 * $a;
```

```
 ?>
```

<br /> At this point, \$a and \$b are equal <br />

So, we change \$a to three times \$a

```
<?php
```

```
 }
```

```
?>
```

# PHP

- PHP overview
- PHP General Syntactic Characteristics
- PHP Output to browser
- Primitives, Operations, and Expressions
- Control Statement
- *Array*
- Function
- File access
- Cookie
- Session
- Form process

# ARRAY

- A PHP array is really a mapping of keys to values, where the keys can be numbers or strings
- Array creation
  - Use the `array()` construct, which takes one or more key => value pairs as parameters and returns an array of them
    - The keys are non-negative integer literals or string literals
    - The values can be anything
    - e.g., `$list = array(0 => "apples", 1 => "oranges", 2 => "grapes")`
    - This is a “regular” array of strings
  - If a key is omitted and there have been integer keys, the default key will be the largest current key + 1
  - If a key is omitted and there have been no integer keys, 0 is the default key
  - If a key appears that has already appeared, the new value will overwrite the old one

# ARRAY

- Arrays can have mixed kinds of elements

- e.g.,

```
$list = array("make" => "Cessna", "model" => "C210", "year" => 1960, 3 => "sold");
```

```
$list = array(1, 3, 5, 7, 9);
```

```
$list = array(5, 3 => 7, 5 => 10, "month" => "May");
```

```
$colors = array('red', 'blue', 'green', 'yellow');
```

[Array.php](#)

# ACCESS ARRAY

- *Accessing array elements* – use brackets
  - `$list[4] = 7;`
  - `$list["day"] = "Tuesday";`
  - `$list[] = 17;`
- If an element with the specified key does not exist, it is created
  - Where??
- If the array does not exist, the array is created
- The keys or values can be extracted from an array
  - `$highs = array("Mon" => 74, "Tue" => 70, "Wed" => 67, "Thu" => 62, "Fri" => 65);`
  - `$days = array_keys($highs);`
  - `$temps = array_values($highs);`

[arraykey.php](#)

## DEALING WITH ARRAYS

- An array can be deleted with unset
  - `unset($list);`
  - `unset($list[4]);` # No index 4 element now
- `is_array($list)`
  - returns true if \$list is an array
- `in_array(17, $list)`
  - returns true if 17 is an element of \$list
- `explode(" ", $str)` creates an array with the values of the words from \$str, split on a space
- `implode(" ", $list)` creates a string of the elements from \$list, separated by a space

[Explode.php](#)



# SEQUENTIAL ACCESS TO ARRAY ELEMENTS

- **current and next**

[accessarray.php](#)

- `$colors = array("Blue", "red", "green", "yellow");`
- `$color = current($colors);`
- `print("$color <br />");`
- `while ($color = next($colors))`
- `print ("$color <br />");`

- **foreach (array\_name as scalar\_name) { ... }**

- `foreach ($colors as $color)`  
`{ print "Is $color your favorite color?<br />";`  
`}`

Is red your favorite color?

Is blue your favorite color?

Is green your favorite color?

Is yellow your favorite color?

# SEQUENTIAL ACCESS TO ARRAY ELEMENTS

- foreach can iterate through both keys and values:
  - `foreach ($colors as $key => $color) { ... }`
- Inside the compound statement, both `$key` and `$color` are defined
  - `$ages = array("Bob" => 42, "Mary" => 43);`
  - `foreach ($ages as $name => $age)`  
`print("$name is $age years old <br />");`

[Keyarray.php](#)

# VIEWING CLIENT/SERVER ENVIRONMENT VARIABLES

- Environment variables [phpinfo.php](http://phpinfo.php)
  - Provide information about execution environment
    - Type of web browser
    - Type of server
    - Details of HTTP connection
  - Stored as array in PHP
    - \$\_ENV

| Variable name | Description                                          |
|---------------|------------------------------------------------------|
| \$_SERVER     | Data about the currently running server.             |
| \$_ENV        | Data about the client's environment.                 |
| \$_GET        | Data posted to the server by the <b>get</b> method.  |
| \$_POST       | Data posted to the server by the <b>post</b> method. |
| \$_COOKIE     | Data contained in cookies on the client's computer.  |
| \$GLOBALS     | Array containing all global variables.               |

**Fig. 26.11** Some useful global arrays.

# PHP

- PHP overview
- PHP General Syntactic Characteristics
- PHP Output to browser
- Primitives, Operations, and Expressions
- Control Statement
- Array
- **Function**
- File access
- Cookie
- Session
- Form process

# USER-DEFINED FUNCTIONS

- *Syntactic form:*

```
function function_name(formal_parameters)
{
 ...
}
```

- *General Characteristics*

- Functions need not be defined before they are called (in PHP 3, they must)
- Functions can have a variable number of parameters
- Function names are NOT case sensitive
- Overloading is not permitted
- The return function is used to return a value
  - If there is no return, there is no returned value

# USER-DEFINED FUNCTIONS

## ○ *Parameters*

- If the caller sends too many actual parameters, the subprogram ignores the extra ones
- If the caller does not send enough parameters, the unmatched formal parameters are unbound
- The default parameter passing method is pass by value (one-way communication) [parameters.php](#)
- To specify pass-by-reference, precede an ampersand to the formal parameter

```
function addOne(&$param) {
 $param++;
}
$it = 16;
addOne($it); // $it is now 17
```

# USER-DEFINED FUNCTIONS

## ○ Parameters

- If the function does not specify its parameter to be pass by reference, you can precede an ampersand to the actual parameter and still get pass-by-reference semantics

```
function subOne($param) { $param--; }
$it = 16;
subOne(&$it); // $it is now 15
```

## ○ *Return Values*

- Any type may be returned, including objects and arrays, using the return
  - If a function returns a reference, the name of the function must have a prepended ampersand
  - `function &newArray($x) { ... }`

# USER-DEFINED FUNCTION

## ○ *The Scope of Variables*

[scope.php](#)

- An undeclared variable in a function has the scope of the function
- If you do want to access a nonlocal variable, it must be declared to be global, as in  
global \$sum;

## ○ *The Lifetime of Variables*

[static.php](#)

- Normally, the lifetime of a variable in a function is from its first appearance to the end of the function's execution  
static \$sum = 0; # \$sum is static

Its lifetime begins when the variable is first used in the first execution of the function, ends when the script execution ends.  
(browser leaves the document in which the php script is embedded)



# PHP

- PHP overview
- PHP General Syntactic Characteristics
- PHP Output to browser
- Primitives, Operations, and Expressions
- Control Statement
- Array
- Function
- File access
- Cookie
- Session
- Form process



# **DATABASE ACCESS WITH PHP AND MYSQL**

# PHP FOR DATABASE ACCESS

- Connect to the MySQL server
  - `$connection = mysql_connect("localhost", $username, $password);`
- Access the database
  - `mysql_select_db("winestore", $connection);`
- Perform SQL operations
- Disconnect from the server
  - `mysql_close($connection);`



## ERROR HANDLING

- All `mysql_` functions return `NULL` (or `false`) if they fail.
- Several functions are helpful in graceful failure
  - `die(string)` - halts and displays the string
  - `mysql_errno()` - returns number of error
  - `mysql_error()` - returns text of error



## ERROR HANDLING EXAMPLES

```
if (!($connection = mysql_connect("localhost", $name, $passwd)))
 die("Could not connect");
```

```
function showerror()
{
 die("Error " . mysql_errno() . " : " . mysql_error());
}
```

```
if (!(mysql_select_db("winestor", $connection)))
 showerror();
```



## BUILDING A QUERY

- Directly
  - `$query = 'select * from wines';`
- Using input information
  - `$winery = $_POST['winery'];`
  - `$query = "select * from wines where winery=$winery";`



## RUNNING A QUERY

- `mysql_query` returns a result handle  
`$result = mysql_query($query, $connection)`
- `mysql_num_rows` indicates the number of rows returned  
`$num_rows = mysql_num_rows($result)`
- `mysql_fetch_array` creates array/hash of result  
`For ($n=0; $n<$num_rows;$n++)`  
`$row = mysql_fetch_array($result)`



## RESULT OF FETCH\_ARRAY

- Contains both numeric and index tags
- Values are duplicated
- Example:
  - Query: `select surname, city from customers;`
  - Row: ( 0=>'Walker', 'surname'=>'Walker', 1=>'Kent', 'city'=>'Kent' );





## PRINTING THE COMPLETE ROW

- By number

```
for ($i=0; $i<mysql_num_fields($result); $i++)
 echo $row[$i] . " ";
```

- By field

```
echo $row['surname'] . ' ' . $row['city'];
```



## AVOIDING SPECIAL CHARACTERS

- When building HTML, characters such as '&' in the data can cause problems
- Function `htmlspecialchars()` replaces all such characters with HTML escapes such as `&amp;`;

```
print(htmlspecialchars($row['surname'] . ' ' .
 $row['city']);
```



## SPECIAL CHARACTERS IN INPUT

- The same problem exists with special characters in input (e.g. ' ')
- PHP switch `magic_quotes_gpc` (default on) inserts backslashes before single & double quotes, backslashes and NULL characters in input data (from GET, PUT and cookie data)
- Use `stripslashes()` to remove the slashes
- Use `addslashes()` to add the slashes if `magic_quotes_gpc` is off



## AVOID DANGEROUS USER INPUT

- Passing user input to other programs opens the door to exploits
  - Eg. `exec("/usr/bin/cal $input")`
  - Generates a calendar (`/usr/bin/cal 2004`)
  - But a malevolent user might send `'2004 ; cat /etc/passwd'` or `'2004 ; rm *'`
- Overlong inputs can also cause problems
- Always clean input
  - `$input = escapeshellcmd($input);`
  - `$input = substr($input,$maxlength);`



## PHP / FORM IN ONE DOCUMENT

- Combine the original form with the PHP document that processes data

```
if empty($regionName) { //parameter provided?
 //produce the <form>
}
else {
 //run the query using data from $_GET or
 $_POST
}
```



## INSERTING INTO A DATABASE

- Collect data from a form
- Validate data (JavaScript, PHP or both)
- Create a query
  - `$query = "insert into customer set cust_id = NULL, "`  
`"surname =\" . $surname . "\" ...`
- Run the query
  - `mysql_query($query, $db);`



## UPDATING A DATABASE

- Query to find item to update
- Present old information
- Collect new information
- Validate
- Construct and run the update query

