Introduction to Unix

- A brief history of UNIX
- Why use UNIX?
- A model of the UNIX environment
- The UNIX file system (directories, commands)
- File attributes (permissions)

A brief history of UNIX

- UNIX is a:
  - multi-user, multi-tasking operating system
  - machine-independent operating system (“portable”)
- the “UNIX” trademark:
  - owned by AT&T
  - passed to the “Unix System Laboratories” (USL)
  - passed to Novell
  - passed to X/Open Company, Ltd. (1993)
  - X/Open + Open Software Foundation (OSF) → The Open Group
  - The Open Group (1996), http://www.opengroup.org/
- So every manufacturer calls it something else!

A brief history of UNIX (2)

- AT&T / Bells Labs (was Lucent Technologies, now Avaya & Alcatel-Lucent)
  - Unix created by two researchers for their own personal use (Thomson & Ritchie, 1970)
  - academic/research operating system
  - Initially, pros: flexibility, extensibility, file sharing
  - Initially, cons: security, robustness, performance
  - easy to use (in comparison with contemporaneous OSes)
  - the first portable OS where "portable" == “recompilable and executable on another architecture”

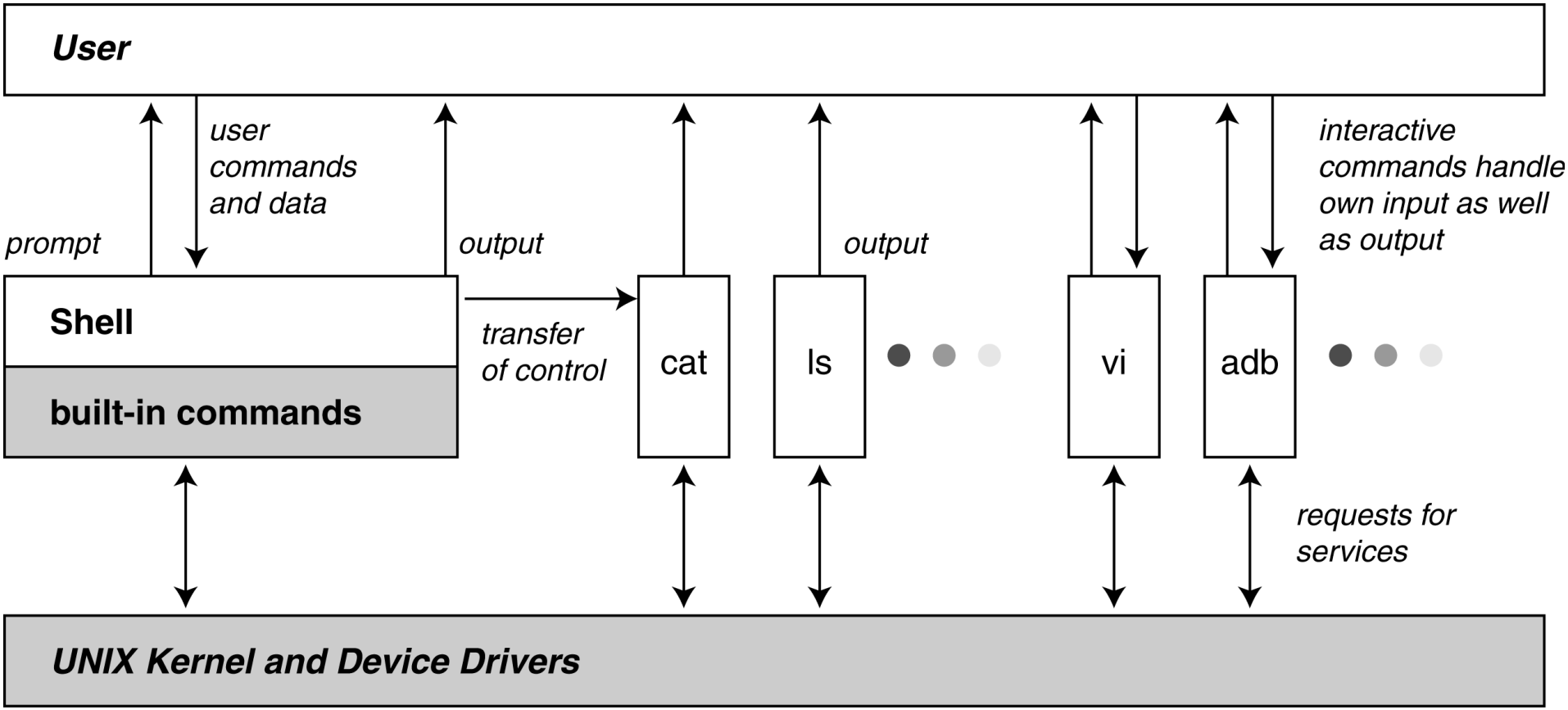
A brief history of UNIX (3)

- Berkeley Standard Distribution (BSD)
  - freeware! (cheap for universities; only paid for distribution cost)
  - first UNIX to include standard network support
  - enhancements to interprocess communication (IPC), job control, security
- many flavours of UNIX in use today:
  - FreeBSD, NetBSD, XENIX, Solaris, SunOS, HP-UX, Linux, A/UX, AIX, Mac OS X
- continues to evolve
  - e.g., Single UNIX Specification (derived from POSIX standard)
- Free Software Foundation and GNU Unix
- Ubuntu – currently the most popular Linux distribution

Why use UNIX?

- multiuser
- multitasking
- remote processing
- safe (stable)
- highly modular
- some versions are free (open→freedom to modify)
- large user community, extensive experience
- “tools are mature”

UNIX model


User
user commands and data
prompt
output
output
interactive commands handle own input as well as output
Shell
built-in commands
transfer of control
cat
ls
vi
adb
requests for services
UNIX Kernel and Device Drivers

shell

- responsible for communication between the user and kernel
- "shell" refers to its position in a diagram of UNIX structure
  - concentric rings
- reads and interprets user commands at the console (or from within a "shell script")
- implements job control
- many shells have been developed:
  - `sh, csh, ksh, tcsh, zsh, bash` …
  - in this course we use the `bash` shell
  - `bash` extends `sh`, the Bourne shell

kernel

- the kernel is the core of the operating system
- the kernel is itself a large and complex program
- clear demarcation between the “kernel” and a “user”
  - to access a computer’s hardware, user must go through kernel
  - “user” must request the kernel to perform work
  - mediated by a command shell (e.g., **bash**), or the system library (compiled application)
- main responsibilities
  - memory allocation
  - **file system**
  - loads and executes programs (assumes a process model)
  - communication with devices (input, output)
  - bootstraps the system
  - …

UNIX filesystem

- "file", "filesystem": **the key abstractions** of the UNIX computing model
- practically anything can be abstracted as a file (devices, programs, data, memory, IPC, etc.)
- mainly responsible for abstracting blocks of information on **physical** storage device (hard drive, flash memory) into a **logical** blocks that user can manipulate
  - maps filenames to block numbers
  - handles block allocations; chains units together
  - provides methods to access data
- facilitates the "multiuser" view of the OS

UNIX filesystem

- arranged as a hierarchy (tree-like structure)
  - organized as a collection of directories; think of a directory as a folder
  - forward slash "/" is used to separate directory and file components (in Windows we use "\")
- the root of the filesystem is represented by the **root-directory**, which we denote by a single "/"

Part of a UNIX filesystem tree