MINIX 3: A Highly Reliable, Self-Repairing Operating System

MINIX-3 design principles

- Introduction
  - computer users
    - have changed with time
    - from: scientists, programmers, engineers
      - **speed very important**
      - **reboots were a natural part of computing**
    - to: children, teenagers, students, ...
      - the electronic model
        - 1-buy device
        - 2-turn it on
        - 3-works perfectly for 10 years

MINIX-3 design principles

- computers don't fit the model
  - complex system
  - crashes and blue screens
  - viruses
  - worms
  - spyware
  - spam
  - ....

MINIX-3 design principles

- need longer MTTF (mean time to failure)
- average user: children, teenagers, student
  - slow worksheet ?
  - slow saving your text file ?
- reliability more important than speed
- OS reliability is poor [Tanenbaum]
  - bugs crash the system
  - bug in a device driver
    - all users are affected
    - loose current work

MINIX-3 design principles

- software bugs are a fact of life...
  - try to anticipate failure
  - deal with failure
  - repair failures
- self repairing systems
- software
  - 1-16 bugs per 1000 lines of code
- reduce size of critical software modules
- microkernel design
  - isolate components from critical software modules

MINIX-3 design principles

- Reincarnation
  - computer failures
    - mainly due to software
  - operating systems are unreliable
  - early designs
    - single monolithic monitor
      - due to slow computers

MINIX-3 design principles

- computer science cycle
  - compiled vs. interpreted programs
    - compiled programs are faster, but more difficult to program and debug
    - compilers: FORTRAN, COBOL PI/1
    - interpreters: Basic, UCSD Pascal

FORTRAN ⟶ UCSD PASCAL ⟶ C ⟶ JAVA

MINIX-3 design principles

- computer science cycle
  - virtual machines
    - developed by IBM mid 1960's (VM/370)
    - CP-67 ran in kernel mode
      - created environment of multiple IBM 360/67 machines
      - upper layer of single user operating system CMS
  - popular again in 2000
    - VMware
    - XEN

MINIX-3 design principles

- computer science cycle
  - protection
  - software vs hardware protection
    - B5000 used software protection (early 1960's)
      - written in type safe ALGOL
    - MMU hardware protection
    - abandoned for many years
    - Microsoft Research
      - Singularity
        - does not rely on MMU
        - written in type safe Sing#

MINIX-3 design principles

- computer science cycle
  - multithreading
  - IBM's multitasking in 1960's
    - tasks shared a memory partition
    - CPU (time slice) shared among tasks
    - used for timesharing systems

MINIX-3 design principles

- computer science cycle
  - microkernel design
    - mid 1980's
      - Amoeba, Chorus, Mach
    - OS running on top of microkernel in user mode
      - message passing for service requests
        - speed vs. reliability
        - Berkeley UNIX running on top of MACH
        - Mac OS X
      - is it back?
      - speed vs reliability

MINIX-3 design principles

- microkernels
  - self organizing OS
    - detect and repair failures
    - kernel bugs are fatal
    - user-mode bugs are not
    - Windows XP kernel
      - 5 million lines of C/C++ code
    - Linux kernel
      - 2.5 millions of C code
    - reduce number of lines in kernel code

MINIX-3 design principles

- move code to user space
  - does not eliminate bugs
  - reduce bugs in critical kernel mode execution
    - smaller kernel --> less bugs
  - bugs will not affect entire system
  - faulty components can be replaced
- MINIX3
  - micro-kernel based operating system
  - fault isolation
  - recovery
    - self-repairing

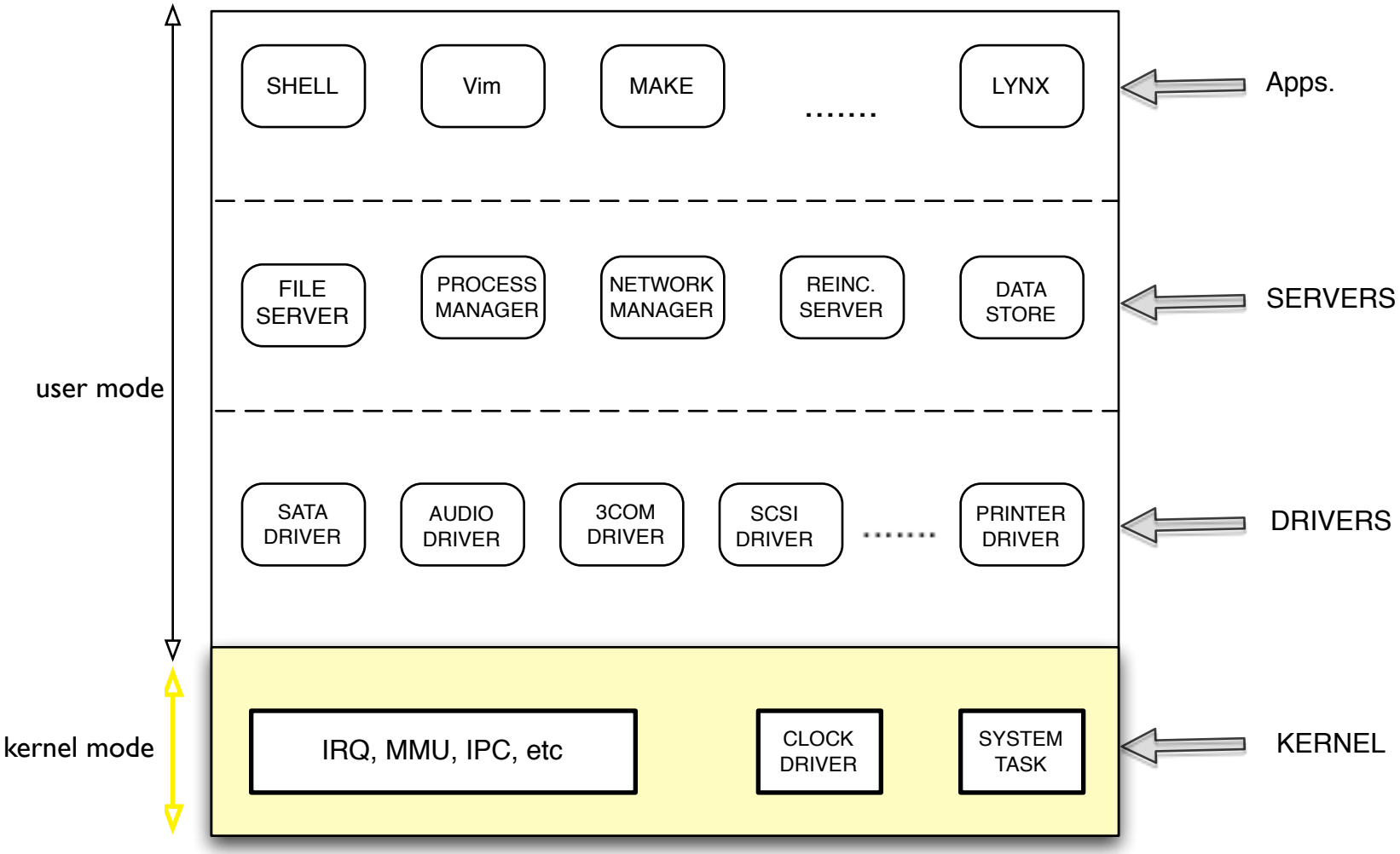
MINIX-3 design principles

- MINIX Architecture
  - goal
    - reliable operating system
    - small trusted microkernel
    - run all servers and drivers as independent user-mode processes
    - monitoring server
      - reincarnation server
        - replace malfunctioning servers
        - shutdown gracefully

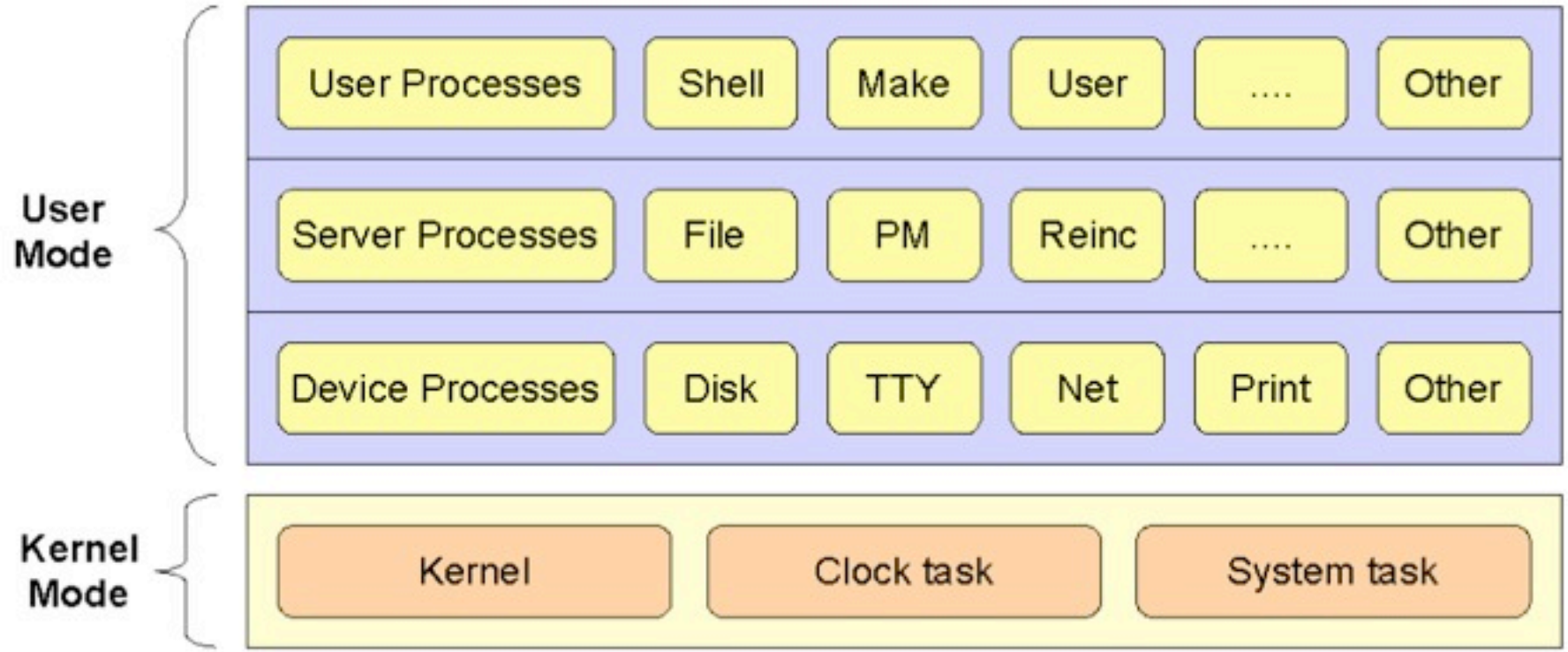
MINIX-3 design principles

- simple algorithms
  - more memory
  - “less” bugs
  - tradeoff between reliability and performance
- microkernel
  - interrupt handling
  - programming the CPU and MMU
  - scheduling
  - interprocess communication
  - kernel calls

MINIX-3 design principles



MINIX-3 design principles



The MINIX 3 Microkernel Architecture

MINIX-3 design principles

- policies
  - part of user-mode OS servers
- processes
  - UNIX processes
    - private address space
      - protected by MMU hardware
    - file descriptors are managed from user-space
  - servers require more privileges than ordinary processes
    - treated equally by kernel
  - all servers and drivers run as independent user-mode processes

MINIX-3 design principles

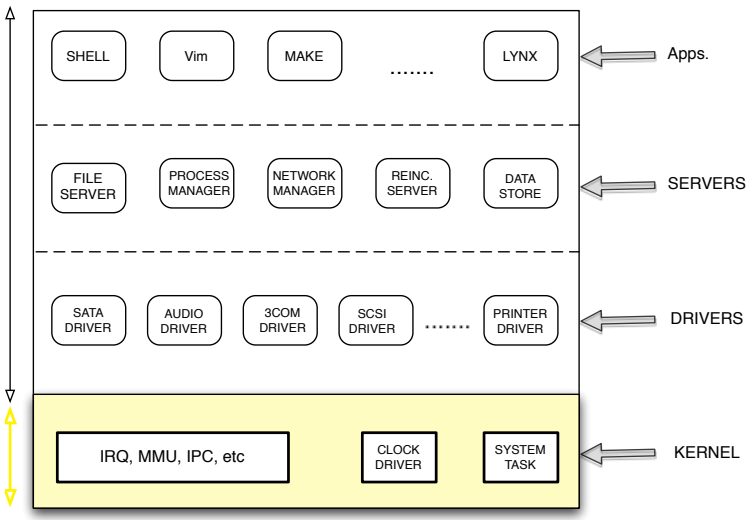
- processes communicate with each other and kernel
  - fixed length messages
  - block on receive/empty send/no receiver
  - kernel copies message from kernel address space to receiver's address space
  - eliminates
    - complex buffer management
    - buffer overflow
  - also has a notification mechanism
    - no blocking

MINIX-3 design principles

- kernel address space
  - clock
    - scheduling
  - system task
    - kernel calls
      - by authorized process
      - low level functions
        - I/O
        - copy to/from kernel address space

MINIX-3 design principles

- device driver layer
  - each a separate process
    - normal user processes
    - protected by MMU
  - scheduled by microkernel
  - send messages and make kernel calls



MINIX-3 design principles

- server layer
  - ordinary processes
  - file server
    - accepts requests from user programs
      - open , read, write, ...normal POSIX system calls

example: read
user process sends fixed-length request message to file server

if data not in cache file server sends a fixed-length message to disk driver to place data in file server's cache

file server makes a kernel call asking system task to copy data to user

- extra overhead:
  - four messages
  - extra context switching

MINIX-3 design principles

- other POSIX functionality
  - process manager
    - fork();
    - memory allocation
    - signals
    - implements policies
    - kernel handles only low-level mechanisms
  - network server
    - TCP/IP stack for BSD sockets

MINIX-3 design principles

- self repairing property
  - reincarnation server
    - manages all servers and drivers
    - monitors system
  - data store sever
    - backup state
    - publish information about system's configuration

MINIX-3 design principles

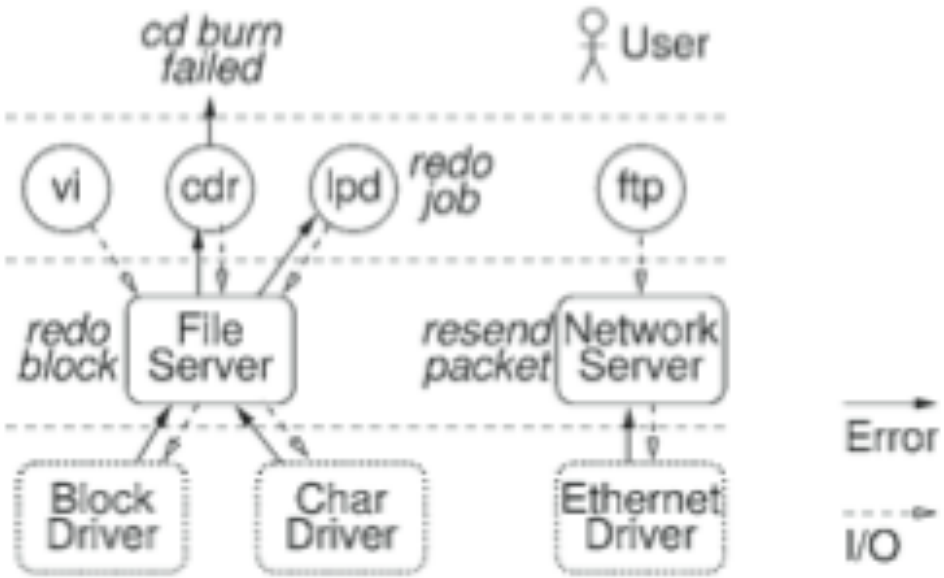
- reincarnation server
  - detect crashes
    - parent/child relation
    - periodic send status request to child processes
  - policy script
    - replace component
    - record event in a log file
    - core dump
    - email administrator
  - system not interrupted

MINIX-3 design principles

- self repairing property
  - problem can be detected
  - restarting component repairs defect
    - transient failures
      - deadlocks, timing issues
    - aging bugs
      - implementation problems that cause failure over time
        - memory allocation

MINIX-3 design principles

- data store aids in recovery
  - state and configuration
- servers are notified
  - if server can't recover
    - error send to application level



MINIX-3 design principles

- Top 10 reliability features
  - ***small microkernel***
    - 4000 lines of code
      - Linux has 2.5 million
  - ***drivers in user space don't eliminate bugs***
    - reduces damage they can cause
    - processes do not corrupt each other
    - at 10 bugs/1000 lines of code
      - MINIX 50 bugs
      - Linux 25,000 bugs,
      - Windows 50,000 bugs

MINIX-3 design principles

- ***drivers don't execute privilege instructions***
  - limits port usage for process
    - driver makes kernel call giving port to use
    - kernel checks at each call
- ***self repairing***
  - reincarnation server
    - crashed or misbehaved servers
- ***synchronous fixed length communication messages***
  - size not a parameter of the IPC
  - size set systemwide at compile time

MINIX-3 design principles

- ***interrupts and messages are unified***
  - drivers are notified when interrupt occurs
    - send a message to driver
    - notification message if busy
- ***bad pointers cannot corrupt memory***
  - servers and drivers protected by MMU
  - encapsulated in user space
- ***no buffer overrun***
  - fixed size messages
- ***infinite loop in drivers will not hang system***
  - reincarnation server will notice infinite loop
- ***restrict drivers and servers to subset of kernel calls***

MINIX-3 design principles

MINIX-3 design principles