TI III: Operating & Communication Systems

Booting and System Services

System Startup,

Bootloader,

Kernel Initialization,

System Services

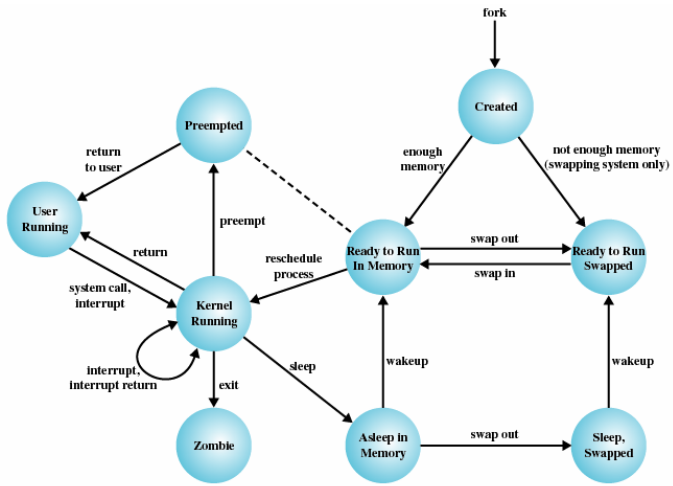


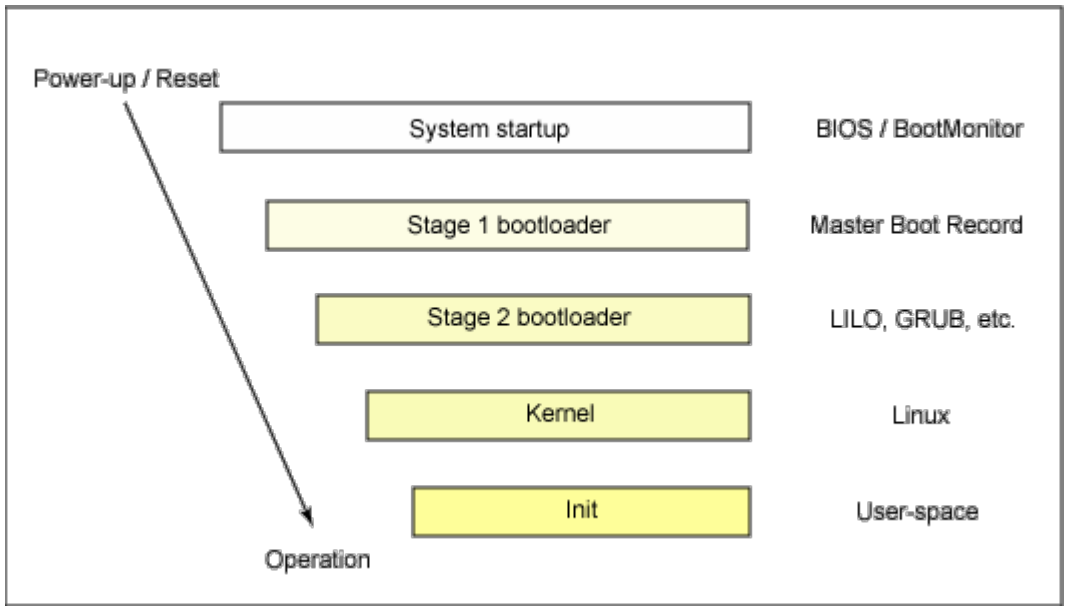
Figure 3.17 UNIX Process State Transition Diagram

Content (1)

Booting Overview

- Booting is the process of starting the operating system when a user turns on a computer.

- Operating system loading and initialization
- Hardware detection and setup



System Startup (1)

1. Power supply switched on
   - Power supply performs self-test
   - ➢ Sends Power Good signal to CPU if OK
2. CPU timer chip receives Power Good signal
   - Stops sending reset signals to CPU
   - ➢ CPU begins operations
3. CPU executes ROM BIOS code at **0xFFFF0000**
4. ROM BIOS performs basic hardware test
   - Beeps on errors (no video support at this point)
5. BIOS checks for adapters with own ROM BIOS
   - ➢ Video adapter initialized:

```
MATROX POWER GRAPHICS ACCELERATOR

MGA Series

VGA/VBE BIOS, Version V2.2
Copyright (C) 1995, Matrox Graphic, Inc.
Copyright (C), LSI Logic Corporation, Inc. 1990-1991
```

System Startup (2)

6. BIOS checks whether “cold-start” or “warm-start”

7. BIOS performs Power On Self Test (POST)
   - No memory check, if “warm-start”
   - Error messages on non-fatal errors
   - Abort and beep code on fatal error

```
*Award Modular BIOS  v4.60PGA, An Energy Star Ally
Copyright (C) 1984-98, Award Software, Inc.

Version 1.15JE33

Cyrix M II/IBM 6x86MX-233 CPU Found

Memory Test: 131072K OK

Award Plug and Play BIOS Extension v1.0A
Copyright (C) 1998, Award Software, Inc.


Press DEL to enter SETUP
11/19/1998 - VP3 - 536B - W877 - 2A5LEF09C - 00
```

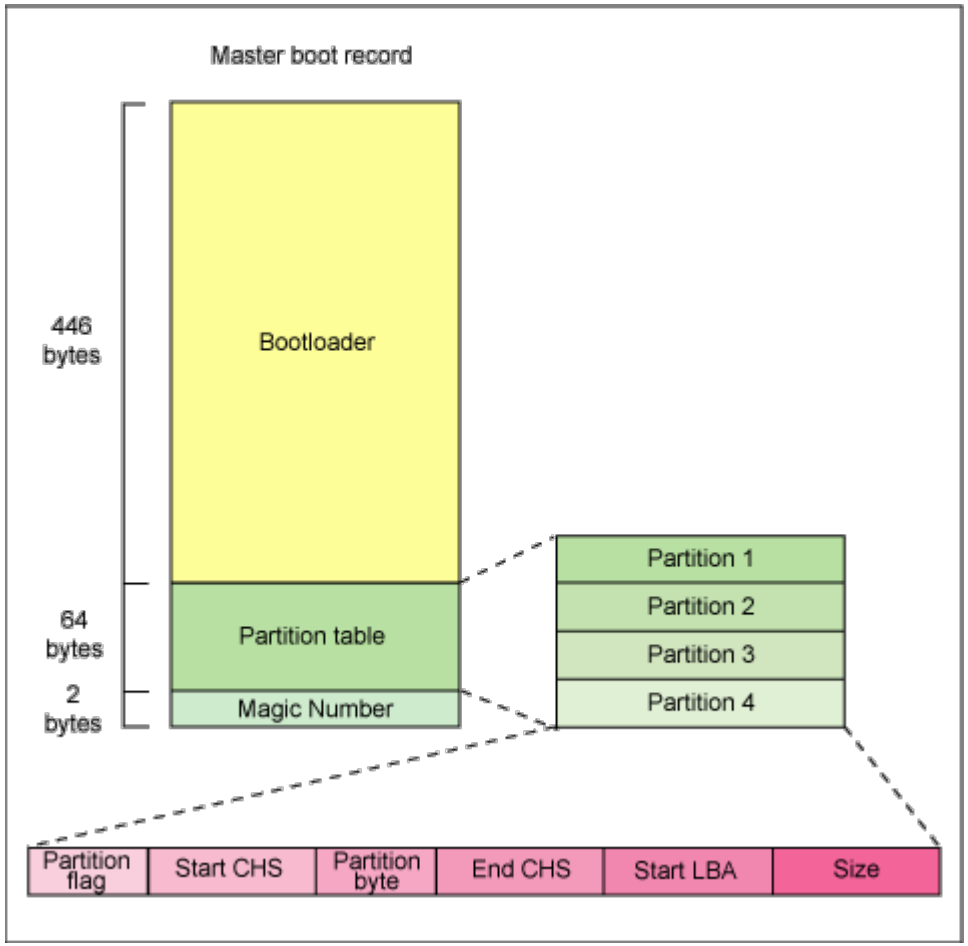
8. BIOS reads system configuration from CMOS

- Defines in which order to check drives for operating system

```
                        Award Software, Inc.
──────────────────────────────────────────────────────────────────────────────
                                      |
| CPU Type     : Cyrix M II/IBM 6x86MX     Base Memory    :      640K  |
| Co-Processor : Installed                 Extended Memory:   130048K  |
| CPU Clock    : 233                       Cache Memory   :     1024K  |
──────────────────────────────────────────────────────────────────────────────
                                      |
| Diskette Drive A: 1.44M, 3.5 in.        Display Type       : EGA/VGA |
| Diskette Drive B: 1.44M, 3.5 in.        Serial Port(s)     : 3F8 2F8 |
| Pri. Master Disk: None                  Parallel Port(s)   : 278     |
| Pri. Slave Disk : None                  Bank 0/1 DRAM Type : None    |
| Sec. Master Disk: None                  Bank 2/3 DRAM Type : EDO     |
| Sec. Slave Disk : None                  Bank 4/5 DRAM Type : EDO     |
──────────────────────────────────────────────────────────────────────────────
```

9. BIOS examines first sector of bootable medium

- Loads Master Boot Record (MBR) into memory



Stage 1 Bootloader

1. BIOS transfers control to bootloader
   - Bootloader checks disk for active partitions according to partition table

2. Bootloader loads stage 2 bootloader from disk
   - Windows: NTLDR
   - Linux: Linux Loader (LILO), GRand Unified Bootloader (GRUB)

Stage 2 Bootloader (1)

Windows / NTLDR

1. Initial bootloader phase
   - Switch CPU from real-mode to protected mode
   - Turns on memory paging
   - Loads file system drivers
2. Operating system selection
   - Load configuration from **`BOOT.INI`**
   - ➢ If more than one OS configured, display choices
3. Allow to set boot options using “F8”

Linux / GRUB

1. Load file system drivers
2. Load configuration from **`/etc/grub.conf`**
3. Display choices of operating systems or kernels, if any
   - Also allows to edit kernel parameters
   - Optional command line shell

Stage 2 Bootloader (2)

4. Hardware detection
   - Run **NTDETECT.COM** to perform hardware detection
5. Configuration selection
   - Display choice of hardware profiles, if any
6. Load kernel from **NTOSKRNL.EXE**
   - Initialize Hardware Abstraction Layer (HAL)
7. Load device drivers of boot devices
8. Pass control to kernel

4. Load kernel image
5. Optionally, load Initial RAM Disk (initrd) image
   - Contains in-memory file system with kernel modules / device drivers
6. Pass control to kernel

Kernel – Windows / NTOSKRNL (1)

Freie Universität Berlin

Phase 0

1. Initialize microkernel and Executive subsystem
   - Disable interrupts
2. HAL prepares interrupt controller
3. Initialize
   1. Memory Manager
   2. Object Manager
   3. Security Reference Monitor
   4. Process Manager

Phase 1

- Enable interrupts

4. Detect additional CPUs
5. (Re)initialialize
   1. Object Manager
   2. Executive
   3. Microkernel
   4. Security Reference Monitor
   5. Memory Manager
   6. Cache Manager
   7. LPCS
   8. I/O Manager
      - Loads additional device drivers
   9. Process Manager
   10. Session Management Subsystem (SMSS)

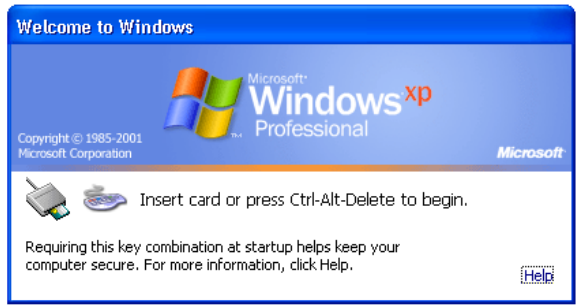
Kernel – Windows / NTOSKRNL (2)

- SMSS runs in user-mode, but is trusted part of the system

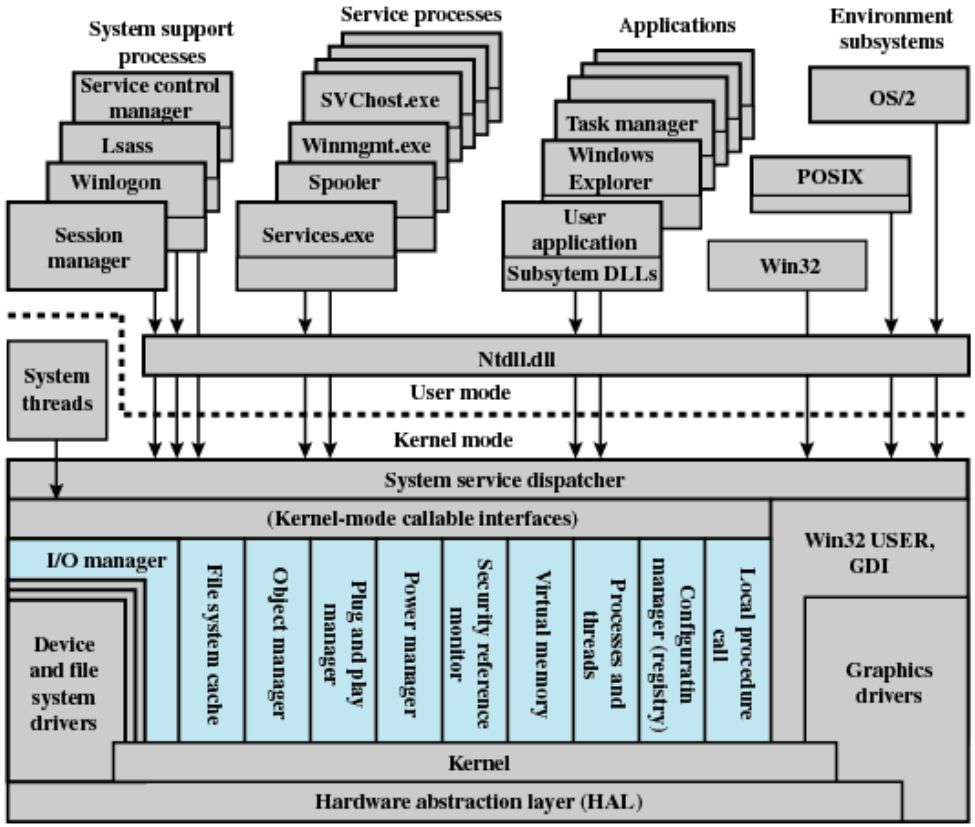
6. SMSS loads Win32 graphics subsystem from **`win32k.sys`**
   - ➢ Switch into graphics mode

7. Services Subsystem loads “Auto Start” services

8. **`WINLOGON.EXE`** and Local Security Authority (**`LASASS.EXE`**) display logon dialog box



Windows 2000 Architecture



Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

Figure 2.13 Windows 2000 Architecture [SOLO00]

Kernel – Linux (1)

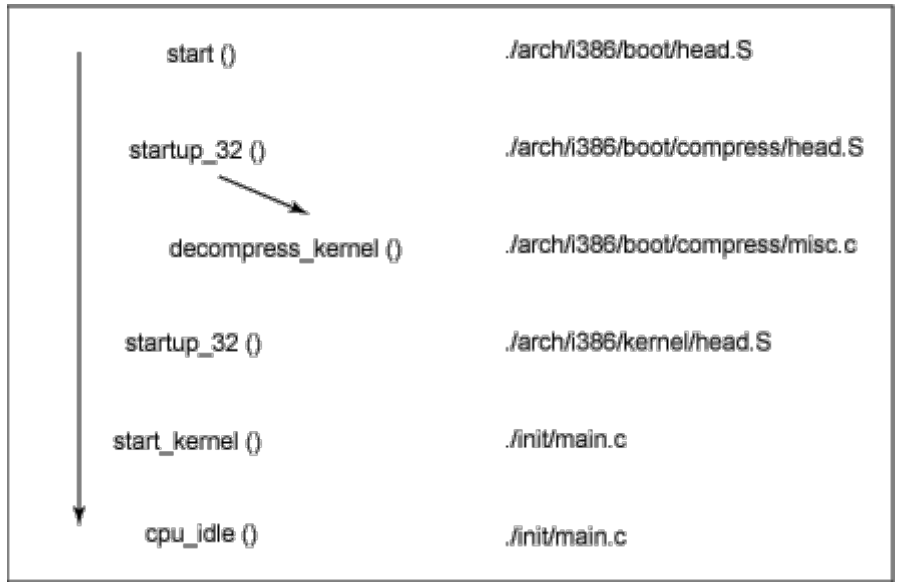
- Kernel image loaded by GRUB is compressed

`start()`:

1. Setup basic environment (stack, ...)

`startup_32()`:

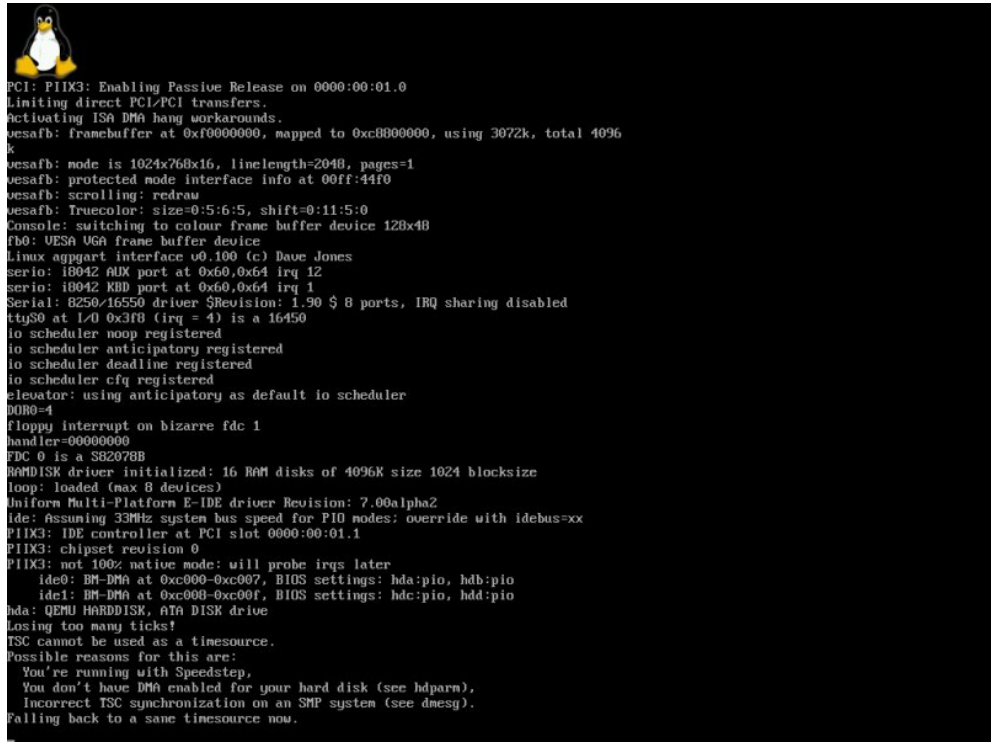
2. Switch to protected mode
3. Decompress kernel image
4. Enable page tables and memory paging
5. Detect CPU and FPU



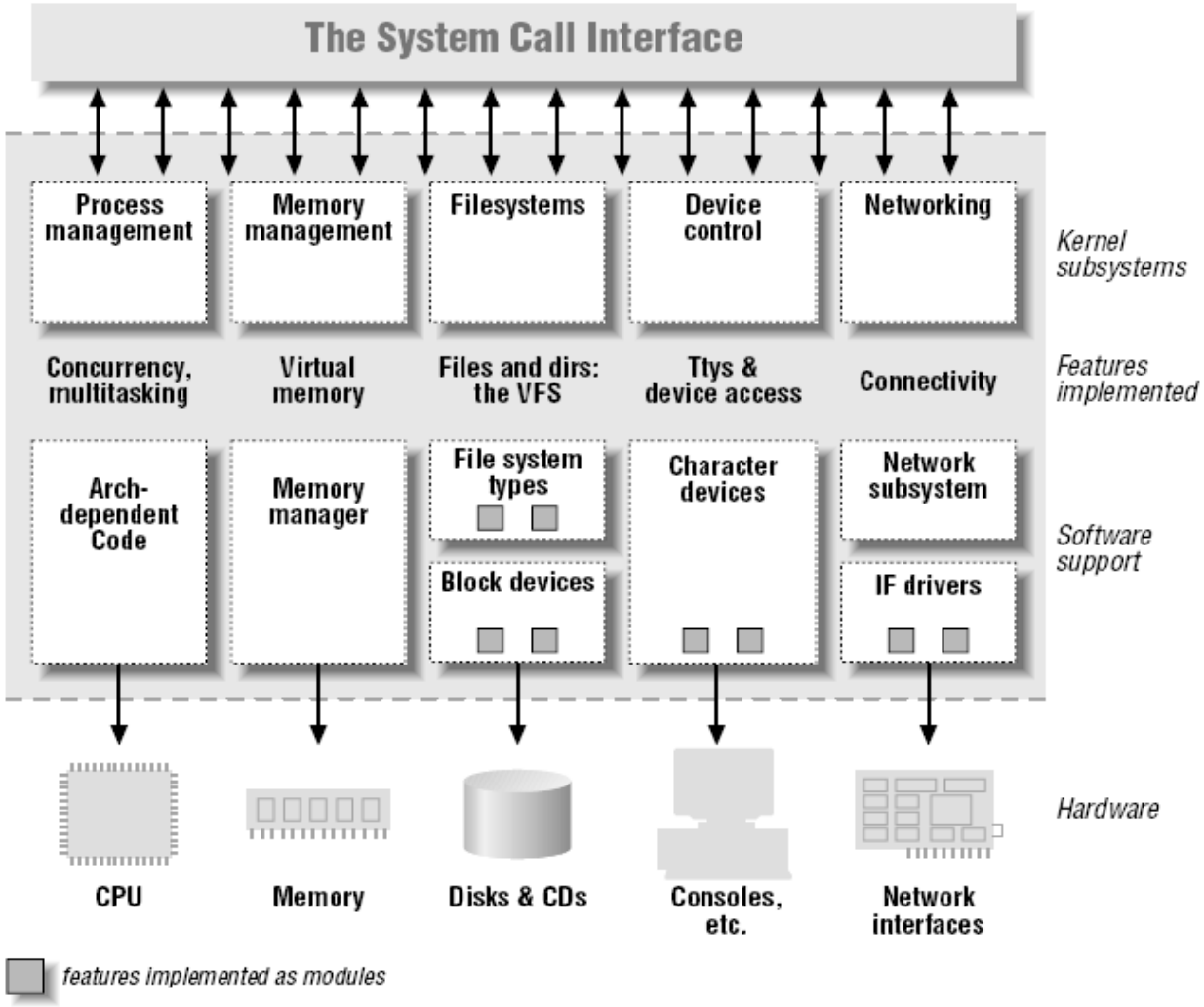
Kernel – Linux (2)

`start_kernel()`:

6. Enable interrupts
   1. Set up interrupt tables
   2. Activate timer interrupt
7. Load initial RAM disk
8. Detect and initialize basic hardware
   - Initialize console
9. Set up kernel subsystems
   1. Buffer management
   2. Signal handling
   3. File and inode management
10. Start **`init`** thread
11. Yield control to scheduler
    - **`init`** sets up remaining subsystems



Linux Kernel Architecture



Linux – SysV Init (1)

1. **`init`** thread executes user-space **`init`** process
   1. Executes one of **`/sbin/init`**, **`/etc/init`**, or **`/bin/init`**
      - Falls back to **`/bin/sh`**, if none of above available
   2. Connects init process to **`/dev/console`** for **`stdin`**
   3. Connects **`stdout`** and **`stderr`** by calling **`dup()`**
   - Leave kernel-mode, drops privileges, releases locks
2. **`init`** performs basic system initialization tasks
   1. Remount root file system with read-write access
   2. Turn on swapping to secondary storage
   3. Mount other file systems
   4. Turn on quota
   5. Set up support for hot-pluggable devices
   6. Configure network interfaces
   7. Mount network shares
   8. Synchronize hardware clock (with external source)

3. SysV Init starts system services according to runlevels

- Typical runlevels (vendor-specific):
  - Runlevel 0: Halt
  - Runlevel 1: Single-user mode
  - Runlevel 2: Not used (user-definable)
  - Runlevel 3: Full multi-user mode
  - Runlevel 4: Not used (user-definable)
  - Runlevel 5: Full multi-user mode (GUI login screen)
  - Runlevel 6: Reboot

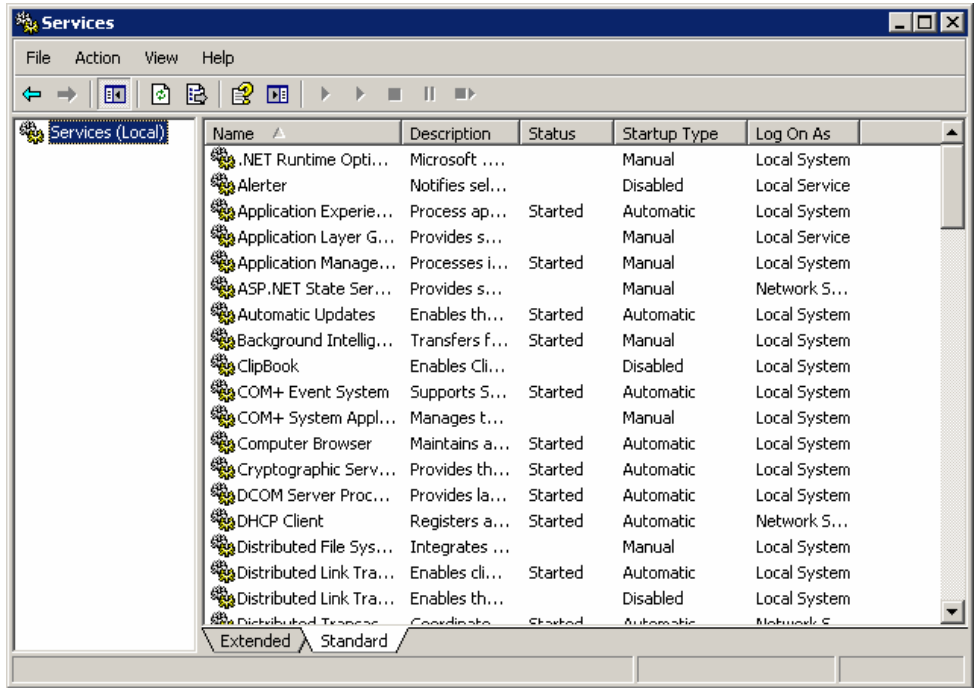
- Default runlevel defined in **`/etc/inittab`**

Typical system services

4. System logging daemon
5. Remote Procedure Call (RPC) server
6. Spam filter
7. ACPI daemon
8. Print spooler
9. Mail Transfer Agent (MTA)
10. SNMP daemon
11. SSH server
12. X Window Server (GUI)

Windows Services

- Service application with interface to be controlled by Service Control Manager (SCM)
- Started at boot time or using Services control applet
  - Runs in background
  - Independent of user
- Run with privileges of LocalSystem user



UNIX Daemons

- System services on UNIX-derived systems
- Typically started at boot time

- Process that runs in background
  - Disassociate from console (if any)
  - Fork a child on start-up, then exit
    - **`init`** adopts the child
  - Set working directory to root directory
  - Close open file descriptors (if any)
  - Set **`stdin`**, **`stdout`**, and **`stderr`** to logfile, console, or **`/dev/null`**
- May store PID in known file for future interaction

System Services: **crond**

Freie Universität Berlin

- Periodically runs commands, referred to as “cron jobs”
- Commands are configured in **/etc/crontab**:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=staff-simpel2@inf.fu-berlin.de

# m h dom mon dow user  command
25 6    * * *   root    test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6    * * 7   root    test -e /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6    1 * *   root    test -e /usr/sbin/anacron || run-parts --report /etc/cron.monthly
#

#tos: der cron job sollte nicht root-mail zuspammen..
0 4 * * * root /etc/simpel/delayed_update >/dev/null 2>&1
#
# vielleich hilft das gegen eingefrorene maeuse, vv 17.6.02
#0 7 * * * root /etc/init.d/gpm restart > /dev/null 2>&1
```

System Services: **syslogd**

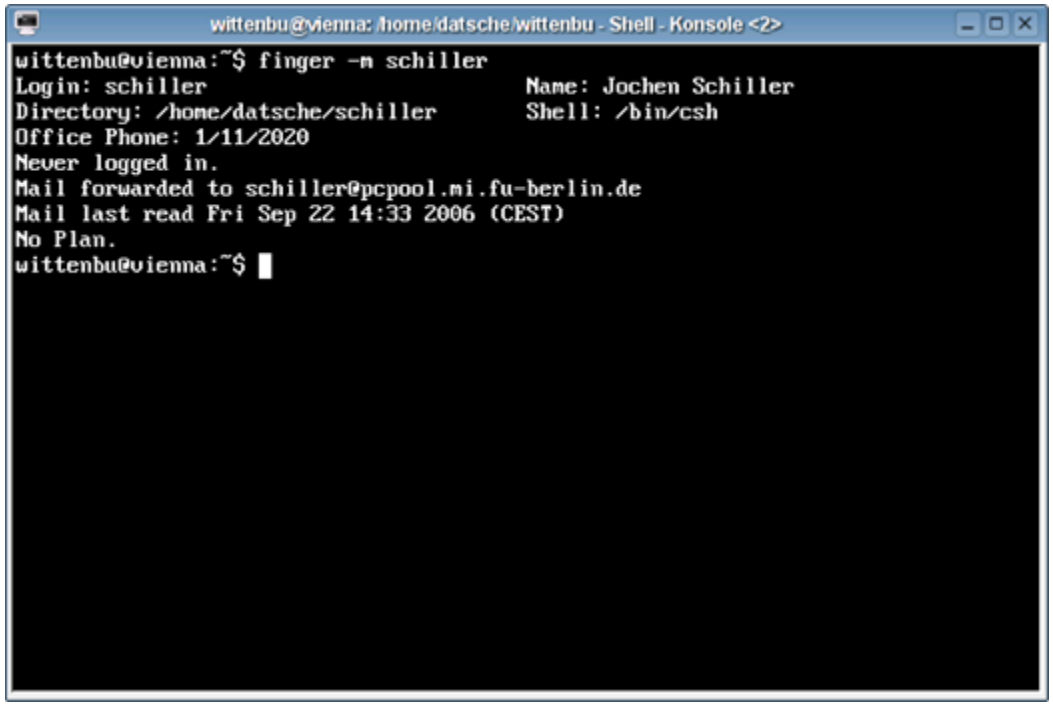
- Daemon for local and/or remote message logging
- Uses Syslog protocol (RFC 3164) to receive messages via UDP or TCP on port 514
- Writes messages to **/var/log/syslog:**

```
Nov 28 01:31:59 vaio syslogd 1.4.1#18: restart.
Nov 28 01:31:59 vaio anacron[4603]: Job `cron.daily' terminated (mailing output)
Nov 28 01:32:00 vaio anacron[4603]: Normal exit (1 job run)
Nov 28 01:36:50 vaio kernel: usb 1-1.1: new high speed USB device using ehci_hcd and address 7
Nov 28 01:36:50 vaio kernel: usb 1-1.1: configuration #1 chosen from 1 choice
Nov 28 01:36:50 vaio kernel: usb-storage: device found at 7
Nov 28 01:36:50 vaio kernel: usb-storage: waiting for device to settle before scanning
Nov 28 01:36:55 vaio kernel:   Vendor: WDC WD16  Model: 00BB-22GUC0       Rev: 0811
Nov 28 01:36:55 vaio kernel:   Type:   Direct-Access                      ANSI SCSI revision: 00
Nov 28 01:36:55 vaio kernel: usb-storage: device scan complete
Nov 28 01:36:55 vaio kernel: SCSI device sda: 312581808 512-byte hdwr sectors (160042 MB)
Nov 28 01:36:55 vaio kernel: sda: test WP failed, assume Write Enabled
Nov 28 01:36:55 vaio kernel: sda: assuming drive cache: write through
Nov 28 01:36:55 vaio kernel:  sda: sda1
Nov 28 01:36:55 vaio kernel: sd 0:0:0:0: Attached scsi disk sda
Nov 28 01:37:03 vaio hald: mounted /dev/sda1 on behalf of uid 1000
Nov 28 01:55:40 vaio hald: unmounted /dev/sda1 from '/media/TREKSTOR-1' on behalf of uid 1000
Nov 28 01:56:23 vaio kernel: UDF-fs: No VRS found
Nov 28 01:56:23 vaio kernel: Unable to identify CD-ROM format.
```

- Separate daemon **klogd** may log local kernel messages

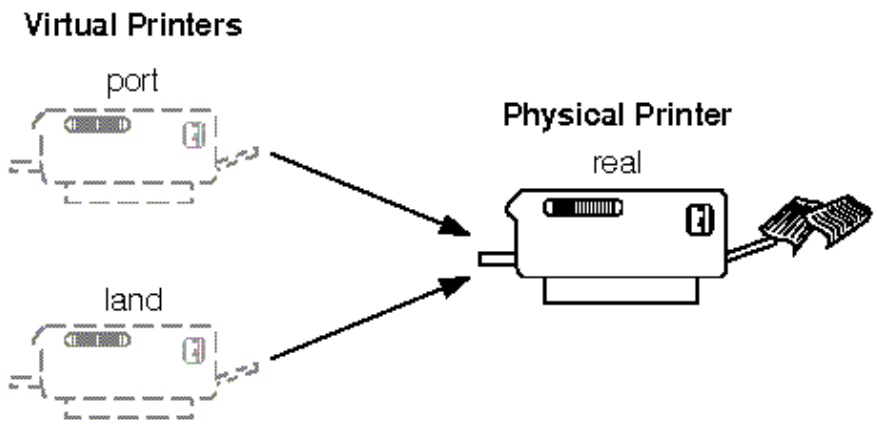
System Services: **`fingerd`**

- Daemon for exchange of user information
- Uses Name/Finger protocol (RFCs 742 and 1288) to process requests sent to TCP port 79
- Provides information from user account database and user's **`.plan`** und **`.project`** files



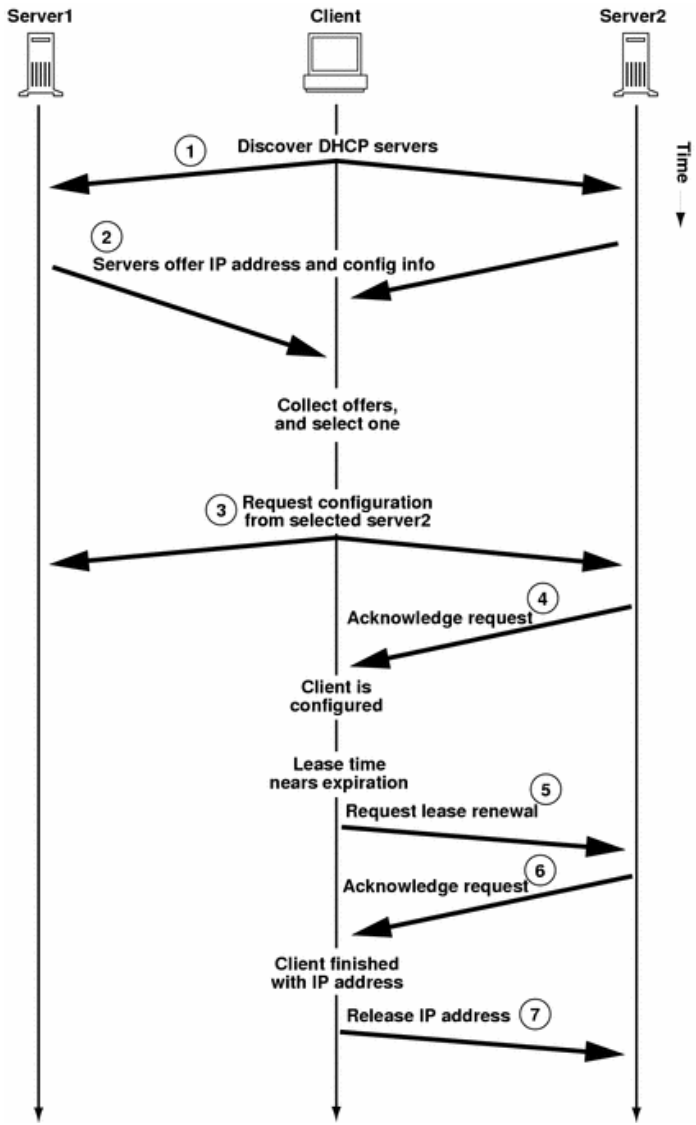
System Services: `lpd`

- Daemon that provides printer spooling
- Manages (multiple) queues for each connected printer
- Controls print jobs of (multiple) users
- Virtual printers may transparently provide different options to real printer



System Services: **dhcpd**

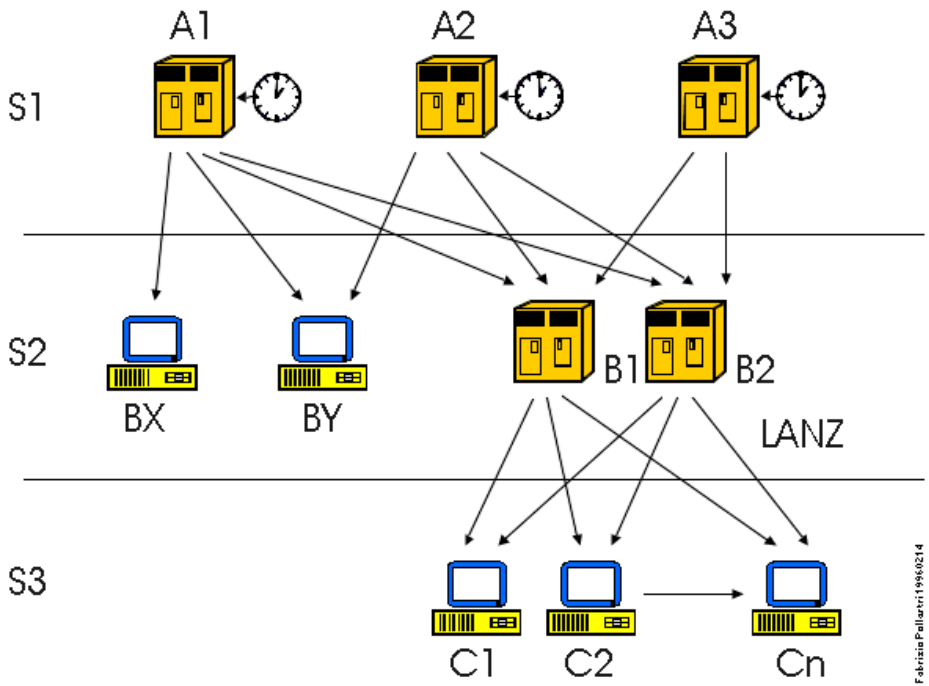
- Dynamic Host Configuration Protocol (DHCP, RFC 2131)
- Daemon acts as server that manages network information (e.g. IP addresses) for local network
- Clients request configuration information at boot time



System Services: **ntpd**

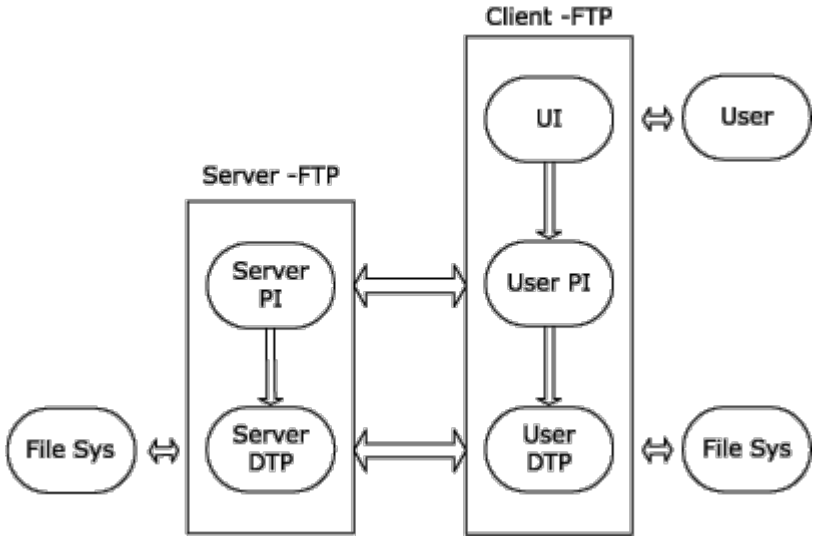
- Network Time Protocol (NTP)
- Synchronizes time of local computer with time servers / other computers
- Daemon listens on UDP port 123

➢ **time.fu-berlin.de**



System Services: **ftpd**

- File Transfer Protocol (FTP, RFC 959)
- Daemon allows remote access to contents of local file system
- Separate connections for commands and data



System Services: **httpd**

- Hypertext Transfer Protocol (HTTP, RFCs 1945 and 2616)
- Daemon listens to TCP port 80 and serves HTML pages / WWW content

Client Request

```
GET /index.html HTTP/1.1
Host: www.inf.fu-berlin.de
```

Server Reply

```
HTTP/1.1 200 OK
Date: Tue, 28 Nov 2006 21:32:18 GMT
Server: Apache
Last-Modified: Sat, 10 Dec 2005 16:31:53 GMT
ETag: "4814c-1914-439b02f9"
Accept-Ranges: bytes
Content-Length: 6420
Connection: close
Content-Type: text/html; charset=iso-8859-1

<html>
<head>
  <link rel="stylesheet" type="text/css" href="http://www.mi.fu-berlin.de/styles/homepage.css">
  <title>Fachbereich Mathematik und Informatik</title>
</head>
<body>

<h1>Fachbereich Mathematik und Informatik</h1>

[...]
```

System Services: **inetd**

- Daemon to manage different Internet-related services
- **inetd** listens on ports of multiple services
- Launches service instance upon packet reception

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#:INTERNAL: Internal services
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
#time           stream  tcp     nowait  root    internal
#time           dgram   udp     wait    root    internal

#:STANDARD: These are standard services.
#telnet         stream  tcp     nowait  telnetd.telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd  -z secure

# the rsync daemon - important: needed by simpel2
#rsync   stream  tcp nowait root /usr/sbin/tcpd  /usr/bin/rsync --daemon
288     stream  tcp nowait root /usr/sbin/tcpd  /etc/simpel/rsync_for_simpel --daemon \
            --config=/etc/simpel/rsyncd-simpel.conf
```

- No system resources used by daemons of rarely used network services