Unix

Marek Kozłowski

Faculty of Mathematics and Information Sciences
Warsaw University of Technology

Outline of the Lecture

Outline of the lecture

Free Libre Open Source Software

- The term *free* means the right to:
  - use,
  - analyze,
  - modify,
  - redistribute

  the software through availability of its source code.
- FLOSS separates authorship and ownership:
  - redistribution of the original or modified software is allowed if and only if the information on original authors and original license is included;
  - the software 'belongs' to everyone.

FLOSS as Mathematics

- For mathematical theorems the authors must be provided while the term ‘ownership’ doesn’t make much sense.
- All people all welcome to:
  - use,
  - prove (analyze),
  - extend (modify),
  - teach (redistribute)

  the theory.

Proprietary Software as Labs Equipment

- Students may use faculty's computers free of charge.
- The computers still belong to the faculty:
  - any unauthorized hardware or software modifications are prohibited,
  - students cannot sell nor even lend those computers to their friends,
  - students cannot eat nor drink when using the computers,
  - students cannot use them for commercial purposes,
  - the faculty can change the terms of use or can revoke the permissions granted.

FLOSS: *Free Speech* vs. *Free Beer*

- The term *free* refers to freedom (as *free speech*), not price (*free beer*).
- *Freeware*, *demo*, *adware*, *trial* and in some cases *shareware* are free of charge but those are examples of proprietary software. Authors/distributors own the software; users buy or receive free of charge the permission to use it. The owners can define (and change later) the terms of use.
- FLOSS software doesn't imply it is free of charge (see the next slides).

Free Libre vs. Open Source Software

- FLOSS is a general term that covers *Free Software* as well as *Open Source Software*.
- The first one focuses on the philosophical and legal aspects (freedoms) it gives to users, whereas the latter one emphasizes the technical aspects.
- Numerous FLOSS licences do exist although most of them are based on the open source *BSD license* or the free software *GNU GPL license*.

Berkeley Software Distribution License

- BSD license states that:
  1. information on authors and license must be included with original or modified code or binaries and other related materials,
  2. the name of the authors cannot be used to endorse or promote products derived from the software without specific prior written permission,
  3. the software is provided *as is* and any express nor limited warranties are disclaimed.
- Note that BSD and derived software can be used as a part of proprietary software.

GNU General Public License

- With GNU GPL author's rights are copyrighted and users' freedom is guaranteed by *copyleft*:
  1. any licensee who adheres to the terms and conditions is given permission to modify the work, as well as to copy and redistribute the work or any derivative version. The licensee is allowed to charge a fee for this service, or do this free of charge,
  2. the software may be distributed only with the source code and the GNU GPL license,
  3. any restrictions on the rights granted by the GPL cannot be imposed on original nor modified code.
- Note that GNU GPL software cannot be bundled with proprietary software. The *GNU Lesser General Public License* (LGPL) has been designed as a compromise: it allows mixing GNU and proprietary software as shared libraries.

FLOSS Doesn't Mean *Free of Charge*!

- The term FLOSS refers to freedom not costs. FLOSS licences allow charging money for:
  - distributing media (selling CDs, DVDs, etc.),
  - educational services,
  - help in installation/configuration,
  - any kind of support,
  - hardware compatibility guarantee,
  - software customization (tailoring to user's needs),
  - ...
- CentOS and RedHat Enterprise Linux are almost the same distributions. CentOS is available for free. RHEL is sold with a wide range of support services.

Outline of the lecture

Operating System Tasks and Components

- *Kernel*:
  - manages computer hardware resources,
  - provides environment for processes: controls process execution, manages shared resources access and provides inter-process communication,
  - manages file systems.
- Processes can communicate with a kernel via a set of *system calls*.
- An interface for users is provided by the *shell* and a set of programs that encapsulate system calls. Userland programs (applications) are started via shell.

Portable Operating System Interface for Unix

- POSIX is an IEEE 1003 standards family that defines system calls along with a shell and basic utilities.
- Apart from the kernel and implementation details all POSIX-compliant systems are ‘the same’ from the users’ point of view.
- POSIX compliant source code is portable between POSIX compliant operating systems.

Single Unix Specification

- SUS standard was initially developed as a substantially cheaper alternative to POSIX.
- Since 1997 common revisions of POSIX and SUS are being developed by Open Group and IEEE (*Austin Common Standards Revision Group*).
- The very latest release is SUSv4 (2008) also known as POSIX:2008 (formally: IEEE Std 1003.1-2008). The previous and commonly referred one is SUSv3 known as POSIX:2001 (formally: IEEE Std 1003.1-2001).

Unix

- For many years *Unix* was a name of a single, proprietary product.
- Nowadays any OS which is certified as SUS compliant may qualify for the name *Unix*.
- Commercial systems like AIX, HP-UX, IRIX, MacOS, Solaris, Tru64, etc. are *Unices*.
- Most FLOSS systems including Linux/GNU distributions and open *BSD systems are highly SUS compliant. Those are commonly referred as *Unix-like* systems.
- Most information presented throughout this course refer to any of those.

Outline of the lecture

Unix Origins and C Language

- 1969: The first version of Unix created at AT&T Bell Labs.
- 1969-1972: Dennis Ritchie (Bell Labs) develops C programming language aimed at writing hardware-independent operating system software.
- 1972: AT&T Unix re-written in C.
- C (not: C++!) is the most natural programming language for Unix.
- Unix system calls are implemented as C functions.

AT&T Unix and Berkeley Unix

- Anti-trust laws (1956) prohibit AT&T from entering the computer business.
- AT&T decide to distribute Unix code free of charge to universities and other research entities.
- Research at University of California in Berkeley results in numerous improvements and serious system redesign. A DARPA grant leads in developing and integrating the TCP/IP stack into Unix.
- Subsequent Berkeley Unix releases: *1.xBSD* (1978), *2.xBSD* (1979), *3.xBSD* (1979) and *4.xBSD* (1980).
- Numerous BSD concepts were included in next AT&T releases: *System III* (1982) and *System V* (1983).

Commercial SysV Unices

- 1983: U.S. Department of Justice settles its second antitrust case against AT&T and breaks up the Bell System. This allows AT&T turning their Unix (System V, in short: *SysV*, *sys-five*) into a commercial, proprietary product.
- Since 80' AT&T lost interest in collaboration with universities.
- System V code was bought by numerous hardware vendors. Based on this code and free BSD code they started producing their own server operating systems: *AIX* (IBM), *IRIX* (Silicon Graphics) *HP-UX* (Hewlett-Packard), *Solaris* (SUN), etc.

*BSD Systems

- Numerous legal processes in early 90's lead to creation of the *4.4BSD-Lite*.
- 4.4BSD-Lite is an open source (BSD-licensed) Unix system based on 4.4BSD but free of any proprietary AT&T source code.
- 1995: The second release (4.4BSD-Lite2) ends the development of Berkely Unix.
- 4.4BSD-Lite code has been adopted by numerous open projects including *FreeBSD*, *NetBSD* and *OpenBSD* that raised in late 90'. Those systems are commonly referred as *BSD systems.

GNU Project

- 1983: *Richard Stallman* starts the GNU Project which aims at creating a „complete Unix-compatible software system“ composed entirely of free software.
- GNU means *GNU's Not Unix!*
- 1983-1985: Stallman retires from MIT (for full independence) and starts the *Free Software Foundation* which leads, sponsors and promotes the GNU project.
- 1989: the first version of the GNU GPL license is published.

GNU/Linux

- In early 90's most of the GNU software is ready although the kernel (*Hurd*) is still incomplete.
- 1991: *Linus Torvalds* releases his GNU-licensed and GNU-compatible *Linux* kernel.
- GNU/Linux is a family of free *nix operating systems based on the Linux kernel and GNU software.
- GNU/Linux distributions include: *Arch*, *Debian*, *Fedora*, *Gentoo*, *Mandriva*, *SuSE*, *Ubuntu* etc.
- The main differences between them concern startup configuration files and software management solutions.

*nix Systems

- The following main branches can be distinguished in the *nix family:
  - proprietary Unix systems based on the AT&T System V code enriched with more or less BSD-derived extensions. The most significant ones include: *AIX*, *HP-UX*, *IRIX*, *Solaris*,
  - proprietary systems based mostly on the BSD code – *Tru64*, *MacOS*,
  - BSD-licensed systems based on the 4.4BSD-Lite code with no AT&T proprietary code, such as *FreeBSD*, *OpenBSD* and *NetBSD*,
  - GPL-licensed and written from scratch *Linux/GNU* distributions.

Outline of the lecture

Unix Principles

- Separation of processing and interface
- Modularity
- Text manageability
- Explicitness
- File system interfaces
- Privilege separation
- KISS (*Keep It Simple, Stupid!*)

Modularity

- *Do one thing and do it well*: simple "bricks" that may be combined into complex shell commands or shell scripts.
- Separation of (micro)kernel, system and applications.
- Multi-layered standard-compliant architecture: one component can be simply replaced with another one that provides the same interfaces and functionality.

Text Manageability

- ASCII files are portable and offer simpler maintenance and better fault tolerance.
- Text-based interface offers more flexibility.
- User interface provided by a command-line shell.
- *Editable Text Configuration* (*/etc*).
- Services started/controlled by invoking shell scripts.
- Most kernel, daemon and application events are logged in the ASCII format.
- Variety of text processing utilities and languages provided.

Explicitness

- Trackable system calls.
- Numerous process control routines.
- Interface to kernel address space and processes (see the next slide).
- Installation and configuration tasks can be performed manually (although installers may be provided).
- Non-critical services and applications should not be automagically installed nor started.
- *Human is always right.*

File System Interfaces

- Stream-based access to files.
- Devices (including terminals) and pseudo-devices seen as (special) files.
- Processes and address spaces can be seen as directories.
- The same, unified mechanism for reading from / writing to streams no matter if those are assigned to files, devices and other processes.

Privilege Separation

- Administrator (*root*) has unlimited privileges to all system resources.
- Each process runs on behalf of and with privileges of some user (*a process owner*).
- Tasks should run with minimal privileges that allow succeeding.
- Pseudo-users with restricted privileges are created for each type of service: they cannot log in to the system nor start a shell but can own processes.
- Files of different types: executables, libraries, configuration files, databases, logs, etc stored in separate directories for simplified privilege management.

Systemd

- System and service management solution designed for Linux only; incompatible with other Unix-like systems.
- Strongly inspired by Windows XP/7 concepts (*svchost.exe*).
- Implemented by most modern Linux distros. For the list of (glorious) exceptions see: `https://nosystemd.org/` .
- Preferred by (inexperienced) desktop users due to "plug-and-play" and "user-friendly", automagical configuration.
- May slightly increase performance in desktop use.
- Causes serious configuration problems, instabilities and critical security issues in more professional usage.
- Violates fundamental design principles of Unix-like operating systems.