Interprocess Communication in Unix

PD Dr. Hannes P. Lubich

Lecture 227-0587-00, ETH Zurich

Lecture Organisation and Administration
Goals of this Lecture

• Understand the kernel of typical Unix systems
• Understand the design and use of the various interprocess communication mechanisms in Unix
• Obtain some practical experience with programming interprocess communication in Unix
• But not:
  – Promotion of Unix as a solution „for everything“
  – Tutorial for specific Unix variants (Solaris, Linux, ...)
• Prerequisites:
  – Knowledge of „C“ or other higher progr. language
  – Know-how about using Unix (shell, tools etc.)

Contents of this Lecture I

• Introduction, History and Development of Unix
• Design of the Unix Kernel
• Process Administration and Programming
• The Unix File System
• Memory Management
• The Input / Output System
Contents of this Lecture II

• Pipes
• Signals
• Sockets
• Message Passing
• Shared Memory
• Semaphores
• Remote Procedure Calls
• TLI, STREAMS
• Future Unix Developments

Organisational Issues

• The Lecturer
  – Hannes Lubich lubich@tik.ee.ethz.ch, http://www.tik.ee.ethz.ch/~lubich
• The Assistants
  – Will be presented during first lecture, see also separate information leaflet

• Lecture Material
  – Lecture Notes: will be distributed during the lecture

• Organisation of Exercises
  – To be solved in groups of 2 students
  – Due every week, „n-2“ exercises to be solved to meet „Testatbedingung“
  – Exercises distributed / solutions submitted during the lecture
  – Solution ideas will be discussed during the exercises

• Organisation of Exams
  – Will be discussed during last lecture
  – Special provisions for exchange students and “Fachhörer”
Lecture Timetable 2004/2005

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic of Lecture</th>
<th>Exercise Distribut.</th>
<th>Solution Submission</th>
<th>Solution Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>Introduction, History and Development of Unix</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>44</td>
<td>Design of the Unix Kernel</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>45</td>
<td>Process Administration and Programming</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>46</td>
<td>The Unix File System</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>47</td>
<td>Memory Management</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>The Input / Output System</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>49</td>
<td>Pipes</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>50</td>
<td>Signals</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>51</td>
<td>Sockets I (Operating System View)</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>52</td>
<td>Sockets II (Network View)</td>
<td>9</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

2005

| 1    | Message Passing                                       | -                   | -                   | -                   |
| 2    | Shared Memory and Semaphores                          | 10                  | 9                   | 7, 8                |
| 3    | Remote Procedure Calls, TLI, STREAMS                  | 11, 12              | 10                  | 9                   |
| 4    | Future Unix Developments, Comments on Exam            | 13                  | 11, 12 (Testate)    | 10                  |
| 5    |                                                      | -                   | 13 (VL)             | 11, 12, 13 (VL)     |

How to Get Information

- The Public Internet
  - Frequently Asked Questions
  - Requests for Comment
  - Search Engines (Google, Altavista, ...)
  - Discussion Groups and Mailing Lists

- At ETH Zurich
  - Web-Server at tik.ee.ethz.ch (see information leaflet)
  - Fellow Students
  - Fellow Students
  - Lecturer

1 – Introduction © Hannes Lubich, 2003–2005
Literature

• O'Gorman: Operating Systems with Unix, Palgrave, 2001
• Bach: The Design of the UNIX Operating System, Prentice Hall, 1986
• Leffler et al.: The Design and Implementation of the 4.3BSD UNIX Operating System, Addison-Wesley, 1989
• Haviland/Salama: UNIX System Programming, Addison-Wesley, 1987
• Stevens: UNIX Network Programming, Prentice Hall, 1990
• Kernighan/Pike: The UNIX Programming Environment, Prentice Hall, 1984
• Libes/Ressler: Life with UNIX, Prentice Hall, 1989
• Herrtwich/Hommel: Kooperation und Konkurrenz, Springer, 1989

Introduction, History and Development of Unix
„UNIX is the answer“

but what was the question, anyway?

The History of Unix

Source: www.opensource.org/timeline.png
“Who is Who“ in Unix

Beginner
- insecure with the concept of a terminal
- hasn’t figured out yet how to get a directory
- still has trouble with typing a RETURN at the end of each line

Novice
- knows that ‘ls’ will produce a directory
- uses ‘sed’ and ‘awk’ with comfort
- uses the editor, but calls it ‘vye’ or ‘six’
- has heard of ‘C’ but never used it
- is wondering how to move a directory
- knows how to read his mail
- is wondering why the person next to him seems to like Unix so much

User
- uses ‘vi’ and ‘nroff’, but inexpertly
- uses ‘grep’ to search for fixed strings
- has figured out that ‘-’ precedes options
- is wondering how to move a directory
- has attempted to write a ‘C’ program and decided to stick with Pascal
- thinks that ‘adb’ is brand new equipment
- knows how to read his mail and wonders how to read news

Knowledgeable User
- uses ‘nroff’ without trouble, and is beginning to learn ‘tbl’ and ‘eqn’
- thinks that ‘grep’ is ‘lost grep’
- has figured out that ‘mv’ will move directories
- has learned that ‘mv’ doesn’t help
- someone has shown him how to write ‘C’ programs
- once used ‘sed’ to do some text substitution
- has seen ‘db’ used but doesn’t use it himself
- thinks that ‘make’ is for whipped

Expert
- uses ‘grep’ when necessary
- uses macros in ‘vi’, uses ‘ex’ when necessary
- posts news at every possible opportunity
- writes ‘csh’ scripts occasionally
- writes ‘C’ programs with ‘cc’ and compiles them with ‘cc’
- has figured out what ‘&&’ and ‘||’ stand for
- thinks that human history started with ‘th’

Hacker
- uses ‘sed’ and ‘awk’ with comfort
- uses undocumented features of ‘vi’
- writes ‘C’ code with ‘cc’ and compiles with ‘cc’
- uses ‘adb’ because he doesn’t trust source debuggers
- can answer questions about the user environment
- writes his own ‘nroff’ macros to supplement standard ones
- writes scripts for Bourne shell

Guru
- uses ‘m4’ and ‘lex’ with comfort
- writes assembly code with ‘cc’
- uses ‘adb’ on the kernel while the system is loaded
- customises utilities by patching the source
- needs device driver source with his breakfast
- can answer any Unix question after a little thought
- uses ‘make’ for anything requiring two or more distinct commands to achieve
- has learned how to breach security, but no longer needs to try

Wizard
- writes device drivers with ‘cat’
- fixes bugs by patching the binaries
- can answer any question before you ask
- writes his own ‘nroff’ macro packages
- is on a first-name basis with Dennis, Bill and Ken

(Unknown Usenet Source)

General Unix Design

Unix and onions have two things in common:
- A layered structure
- If you look under the shell, you will burst into tears

(Unknown Usenet Source)
The Unix File System

```
/proc             bin                etc
usr               vmunix
dev

23009     sh     date     cc     passwd     src     bin     tty01     sd0a

cmd

date.c     who.c

ls

cd

pwd
```

The "C" Programming Language

1973: Thompson, Ritchie: BCPL - NB - C

1975: Ritchie: stabile Version von C in Unix V6

1978: Kernighan/Ritchie: "The C Programming Language" ("K&R C")

1979: Unix V7 (K&R + Enumeration Types): results in "V7 C"

1980: Unix 32V (V7 C + Extensions): results in "UNIX C"

1986: ANSI C, C++

1990 - C#, G, G++, ...

- C has not been created by programming language designers:
- Syntax and semantics of some language elements feel "strange"
- C is not object-oriented
- Hardware properties ("bits and bytes") are visible in the language:
- portable programming requires good C language skills
- large team programming requires a lot of discipline and control
Unix and „C“

- Close relation between Unix and C:
  - Unix system call interface and system libraries designed in C
  - Kernel and tools/utilities are mostly written in C
- C is a good compromise between a system language and a high-level language, allowing access to „bits & bytes“, as well as structured and modular programming -> well-suited to program Unix systems
- C is currently the most commonly used language for system programming
- C is available on all Unix platforms, as well as on may other operating systems
- Other languages (e.g. Fortran, Pascal) can be translated into C

Use of „C“ within Unix

- Command Interpreter (Shell, GUI, ...)
  - 90 - 95 %
    - Unix Tools ( > 200)
      - ca. 350000 LOCs C
    - C-Libraries
  - Application Programs
  - 5 - 10 %
    - Kernel: System calls
      - ca. 10000 LOCs C, ca. 1000 LOCs Assembler Code
    - Hardware
Why did Unix Succeed?

1. Operating system and tools are written in a High Level Language
   ‣ Portability
2. Simple, easy to exchange user interface (shell) geared towards developers
3. Programs can work as filters / building blocks and can be dynamically combined into complex applications
4. Hierarchical, extensible file system, that can work locally and over networks
5. Simple file format (byte stream)
6. Simple, standardised periphery interfaces, mapped onto the file system
7. Multi-user-/process system, works on everything from mainframes to PDA
8. Optimal use of hardware properties shielded from end users
9. Supports a variety of programming environments besides C / C++
10. In comparison to other OS, Unix is easy to use, administer, and program.
11. Strong growth via Linux and Open Source Community

Unix Market Share and Linux Maturity

Ovum Studie 1996 zur Verteilung von Lizenzumsätzen für Betriebssysteme an Arbeitsplätzen mit betriebswirtschaftlicher Anwendungssoftware in Europa (Angaben in Mio ECU)

Linux Infrastructure, Virtual Hosting and Web Servers
(e.g., File, Print, Cache, Firewall)

Linux Three-Tier Applications

Linux Four-Ways

Complex, MC Workloads

Modular Computing, System and Storage Management

Visibility

Technology Trigger

Peak of Inflated Expectations

Trough of Disillusionment

Slope of Enlightenment

Plateau of Productivity

Maturity Time line

Gartner Group Linux Hype Cycle Q1/2003

As of January 2003

MC = mission critical
RISC = reduced instruction set computer

Key Time to Plateau:

Less than 2 years
2–5 years
More than 5 years
Summary: First Unix Assessment

- Unix is one of the most preferred industrial software development environments (rapid prototyping, silent programming, toolbox approach, networking support etc.)

- Unix will remain widely used on many hardware platforms from PDAs to PCs and mainframes, Unix market share will remain high, due to Linux adaptation by software industry

- Other operating systems have “borrowed” heavily from Unix

- Most programs developed on Unix are easily portable

- **But:**
  Unix is not perfect: Access control, administration, version “jungle”, not well-suited for beginners $\$\$ ("rm -r -f /*")