Lecture 1:
“a small yet powerful operating system”

Unix History and Fundamentals

Introduction

- “Technically, Unix is a simple, coherent system which pushes a few good ideas to the limit.”—Sunil Das
- “The greatest virtues of Unix, in my opinion, are those that emerged as a result of the way that it developed.”—Peter Salus
- “The growth and development of Unix is an exciting sociological tale. . . . The nature of the individuals and their interactions is what made Unix vital.”—Peter Salus
- “UNIX is simple and coherent, but it takes a genius (or at any rate, a programmer) to understand and appreciate its simplicity.”—Dennis Ritchie
- The history of Unix is a story of intrigue, adaptability, desire, cunning, intellectual honesty, and the pursuit of freedom and happiness.—Mark Shacklette

“it was the summer of ‘69”

- The Multics (Multiplexed Information and Computing Service) operating system was being developed jointly by the Computer Research Group at BTL and Fernando Corbato’s MAC (Multiple Access Computers) project at MIT and GE which provided the hardware. Multics:
  - Was based on CTSS (Compatible Time Sharing System) at MIT
  - Was to deliver *multi-user multitasking* support for 300 simultaneous users in an *interactive* (non-batch) system
  - Ran on a powerful machine: a GE 645, a 36-bit machine that executed at about .435 MIPS
    - This was about 30% faster than the original IBM PC (808[6,8]), and a 1000 Mhz Pentium III today is about 2000 times faster than the GE 645
    - The GE645 took about an hour to boot Multics

Space Travel and New Ideas

- Ken Thompson (Berkeley EE) was at this time:
  - designing a file system to support Multics,
  - playing his “Space Travel” game
- Multics had a few new ideas:
  - Virtual Memory for each user
  - Shared runtime libraries
  - Notion of making the “user interface” (shell) a *user program* instead of part of the kernel
  - First Hierarchical file system (novel concept in 1964)
  - Write the OS in a *portable* high level language: PL/1
- According to one tongue-in-cheek oral tradition, Multics was also an acronym for “Many Unnecessarily Large Tables In Core Simultaneously”

The Multics Triumvirate

- After tuning, a Multics machine could support up to 30 users per processor, but performance was somewhat erratic
- In addition, Space Travel's performance on the GE645 was less than stellar, and it cost Thompson's department $50-$75/hour to play (timesharing in 1969 remember!).
- But the triumvirate had different ambitions for Multics:
  - GE wanted Multics as an operating system on which to sell GE computers
  - MIT wanted Multics to advance the state of the art in operating systems research
  - BTL just wanted a good computer on which to get some work done
- But BTL management felt it was costing more than it came to

“the toy was gone”

- In the end, BTL decide to pull out of the Multics effort in the spring of 1969.
- Multics development did continue under the auspices of Honeywell (which had purchased GE’s computer group), but Corbato and MIT pulled out of the effort finally in the mid 1970’s.
- The last Multics machine was shut down in October of 2000 (at the Canadian Department of National Defense)
- Meanwhile, back at BTL, Ken and his group responded by petitioning the management at BTL to purchase a new DEC 10 for about $120,000. Why? The were asked. So they could work on another operating system that looked *a lot like Multics*….

What Part of "No" Don't You Understand?

- Their repeated requests were denied.
- Ken Thompson's wife took their new baby out to California to see his parents, and so he had a month "free".
- Ken and Dennis Ritchie (Harvard math) borrowed a DEC PDP7 from another group (that had already paid $72,000 for it), and Thompson used that month to write a new operating system, a shell, an editor, and an assembler.
- He based the hierarchical file system on the Multics design, implementing it overnight
- In one man-month, a new, albeit rudimentary, multiprocessing, multi-user operating system was written
- As a play on words, Peter Neumann (or Brian Kernighan?) called the new system "UNICS", for UNIplexed Information and Computing Service, as a pun on the somewhat hubristic ambitions of Multics.

Small is Beautiful

- But the 18-bit PDP 7 was a much smaller and slower machine than the GE 645, it had only 8K of user memory and would only support 2 simultaneous users (physical limit).
- This introduced a necessity to think small, and drove a fundamental tendency to write a number of very small utilities that followed the philosophy of “do one thing, and do it well”.
- Dennis Ritchie had ported “B”, a shortened BCPL (Basic Compiled Programming Language) over to UNICS, which was too small to support a large high level language compiler such as FORTRAN or PL/1. In fact, “B” ran as purely an interpreter generating intermediate code.
- Thompson ported over Space Travel onto the PDP 7.

The “Setup”

- The Small is Beautiful philosophy was driven by necessity, and can even be seen in the cryptic and abbreviated names for the various programs: cp (copy), rm (remove), ls (list), cat (catenate), dc (desk calculator), ttt (tic tac toe, a game, Belle in 1980) and ed (line editor). (Many names, like ‘ls’ and ‘pwd’ were literally inherited directly from Multics)
- Remember also that early “monitors” were 30 character/second teletype devices, and each character transmitted could take some time.
- Ritchie, McIlroy, and Ossanna needed a new computer but had no money (and Thompson needed a better platform for Space Travel)
- Luckily, the Patent Department at BTL had some money and needed a new text processing application

The Sting I

- In the summer of 1970, Joe Ossanna suggested they commit to delivering this new text processing system similar to “runoff” for the old CTSS system)
- This would give them the excuse to argue for a new computer (not for a new operating system, but for the patent office’s new text processing system)
- So they got their hot new computer, a 16-bit PDP 11/20 with a whopping 64K of memory (extensible to 256K) and a .5 Megabyte disk for a cost of about $65,000
- But the new machine for the text processing system needed a new operating system, and so they quietly ported UNIX over onto it
- Ritchie and Morris “borrowed” some old code from another system that did text processing, and hacked together a text processing system literally overnight.

The Sting II

- So, after Multics, they couldn't get anyone to buy them a decent computer to write an operating system, but they got instead a great system on which to write a text processing system, which they called "roff", after "runoff".
- Unix was back in business, and had delivered its first system, a text processing system, to the Patent Department at BTL.
- AT&T UNIX Version 1.0 was officially released on 11/3/1971 (read the last sentence of the Programmer's Manual)
- Soon, the Patent Department income provided them a new PDP 11/45 with 256K of memory for further development.

AT&T Unix - First Edition 11/3/71

- Contained around 60 user commands, including chmod, chown, chdir, cmp, date, cp, db, df, du, ld, ln, mv, od, pr, roff, to name a few.
- This constituted the core of Unix, only the concept of “pipes” was missing.
- Edition 2 came out shortly after in June of 1972, and its preface mentioned 10 users of Unix.
- Interest quickly spread through universities, and the list of 16 users for version 3 reads like a university who’s who: Harvard, Berkeley, Columbia, Princeton, Stanford, Johns Hopkins. (Not MIT, why?)
- So what would any red-blooded capitalist American company do with such a promising product?

Here Come da Judge

- But there was a slight problem….
  - 1956 Antitrust decision (from Truman's original 1949 suit) specifically prohibiting AT&T and Western Electric from profiting from anything that didn't directly have to do directly with providing telephone service
  - Did Unix provide Telephone service? No.
  - Could AT&T therefore sell and profit from Unix? No.
- So AT&T decided to license the software *and it's entire source code* to educational institutions for a nominal fee, to cover packaging, and released it w/o support to any interested universities

Edition 3, 2/73

- Edition 3 introduced the C Programming Language, in February of 1973, and the concept of *pipes*.
  - A pipe in Unix is simply a way to connect the *output* of one program and use it automatically as the *input* to another program.
  - Thompson rewrote a number of utilities (again, pulling an all-nighter) to accept *input* directed to it from another program. Such programs were called a *filters*.
  - Another driving force in the creation of pipes and filters was that writing a single large monolithic program probably wouldn't have fit in a 64K process space anyway!

Edition 4, 1973

- C was derived from B which was derived from BCPL, which was derived from FORTRAN.
- Both BCPL and B (and therefore C) contained the notion of pointer indirection
- By the fall of 1973 Unix could support up to 33 simultaneous users issuing over 13,000 commands a day, and was in use by dozens of institutions.
- AT&T Bell Labs had an accidental hit on their hands.
- In November of 1973, the fourth edition (System Four) was released, and Unix itself was rewritten entirely in C, thus making the operating system highly portable to new machine hardware.

Bill Joy and The Berkeley Connection

- In 1975, Ken Thompson was invited to Berkeley to teach an operating systems course. He taught a course on … Unix.
- While at Berkeley, he ported Unix Version 6 onto the Berkeley CS department's new PDP 11/70, which provided a 32 bit system with up to 4Meg of memory addressing
- In 1979 Bell Labs also ported Unix Version 7 over onto a VAX (Virtual Addressing Extension), which supported a 512 Meg maximum memory and virtual memory
- Berkeley took AT&T's VAX port and put it on a brand new VAX 11/780, which had 2 Meg of memory installed.
- Bill Joy brought to Unix the vi editor, termcap, the C shell and command history, job control, and most importantly, the Berkeley Software Distribution (BSD) version of Unix.

The Network

- The ARPANET project of the DoD's Advanced Research Projects Agency was begun in 1969, the same year Ken Thompson was playing Space Travel.
- ARPANET was designed to have an intelligently reconfigurable network that could survive multiple nodes being "taken out" simultaneously. It was a Cold War network.
- Bill Joy helped convince DARPA that Unix, not VMS, should be the OS of choice, because it was portable and wouldn't be tied to a particular hardware platform.
- Thus Unix was chosen as the core operating system by DARPA running on the VAX, in preference to it's native, default OS (VMS)

Berkeley Software Distribution

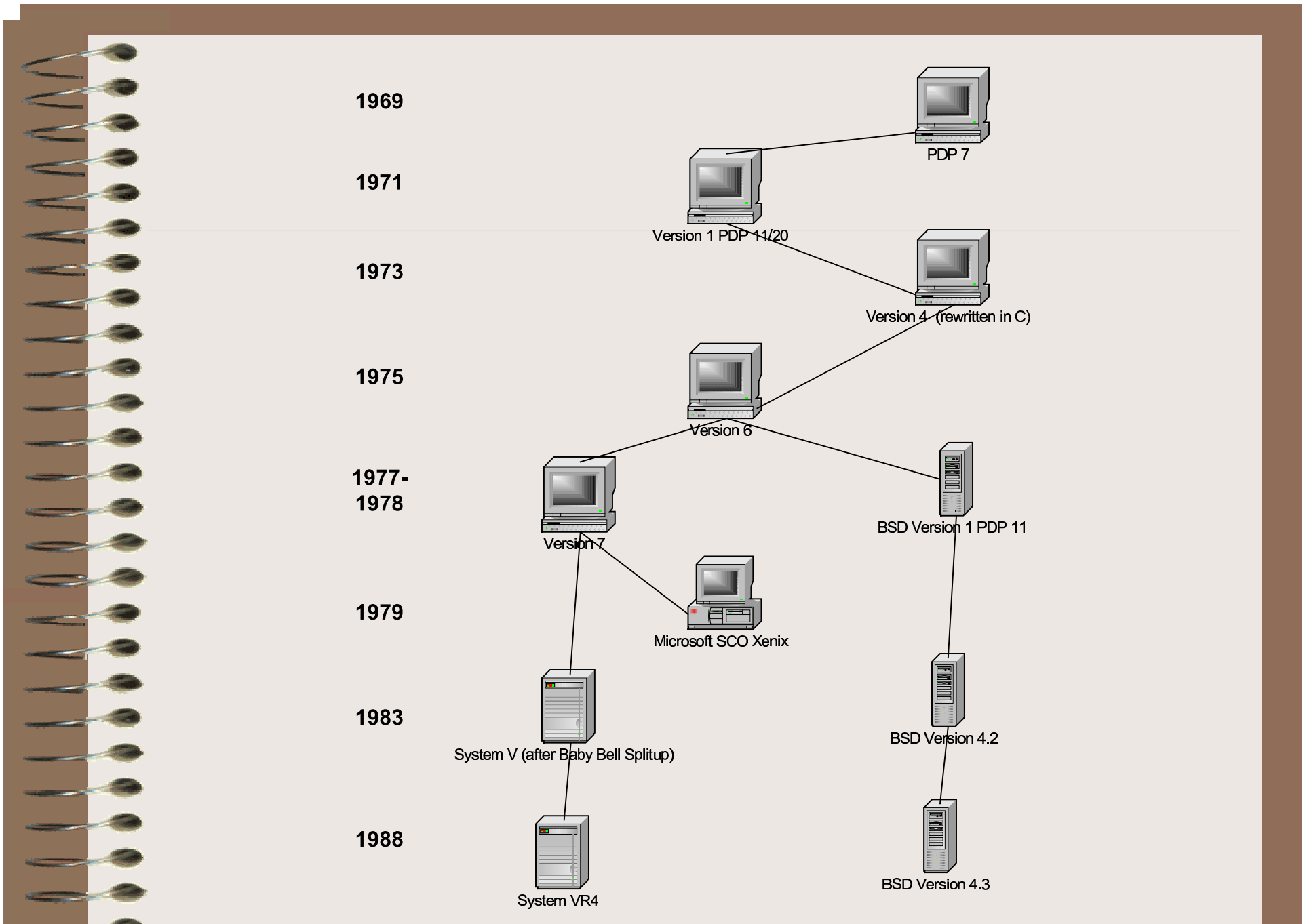
- Bill Joy founded the Berkely Software Distrubtion while working for the Computer Systems Research Group (CSRG) at Berkeley
- BSD 1.0 was released in early 1978 for a license payment of $50.00 for the PDP 11/70.
- BSD Version 3 was released in 1979 and ran on the VAX 11/780 (based on AT&T Version 7 for the VAX)
- BSD Version 4.1a in late 1981 introduced Berkeley Sockets and Version 4.2, in September 1983, introduced networking to Unix in the form of TCP/IP stacks.
- BSD 4.3, June 1990, introduced Network File System (NFS).

BABEL:
The Commercialization of Unix

- In 1982, the Justice Department concluded its long dispute with AT&T by dissolving Western Electric and breaking up AT&T into independent phone-specific companies (Baby Bells), which allowed AT&T to enter the computer business—like sell software
- This meant several things:
  - AT&T could now profit from UNIX
  - UNIX was now protected intellectual property, thus John Lion's famous commentary on Version 6 because a copyright infringement and its distribution as photocopies had to go "underground"
  - UNIX now cost real money, rising as high as $200,000

BABEL:
The Commercialization of Unix

- In 1982, Bill Joy left Berkeley to form SUN Microsystems
- Other heavy-hitters entered the Unix market: IBM, Hewlett-Packard, Digital, Silicon Graphics, etc.
- Unix and its software went through the roof in terms of price
- As a result, free open-source movements began, most notably with Richard Stallman and the Free Software Foundation's GNU project
- Each vendor had its "flavor" of Unix. Efforts, such as POSIX (Portable Operating Systems Based on Unix) were developed to enforce some conformity among the various competing versions


1969
1971
1973
1975
1977-
1978
1979
1983
1988
PDP 7
Version 1 PDP 11/20
Version 4 (rewritten in C)
Version 6
Version 7
BSD Version 1 PDP 11
Microsoft SCO Xenix
System V (after Baby Bell Splitup)
BSD Version 4.2
System VR4
BSD Version 4.3

Unix Core Philosophy Summary I

- Each program should do one thing and do it well. If you need something new, add a new simple program rather than bloat existing programs with new features.
- Allow the output of every program to be the input to another independent program. Don't clutter output with extraneous information. Avoid columnar and binary formats for I/O. Don't insist on interactive input.
- Design and build software, even operating systems, to be tested early on in the development cycle. Throw away the clumsy parts and rewrite them.

Unix Core Philosophy Summary II

- Favor general-purpose software tools, *even* if you have to take the time to stop and write the tools yourself.
- Honesty: Admit your bugs, even advertise them so that others won't trip over them
- Do not assume, a priori, that your users are idiots
- When forced to decide between portability and speed, always opt for portability (speed will come, Grasshopper)
- Give users *choice* (as in shell, editor, etc.)
- Provide online help (from Multics)
- Provide source code for freedom and empowerment

(cf. McIlroy, Pinson, Tague, "Unix Time-Sharing System Forward", *The Bell System Technical Jounal*, July -Aug 1978; 57.6.2, p. 1902)

The Unix File System

- Unix is Pro-Family (Hierarchical)
  - Parent, child, current directories
- Absolute pathnames (start at root)
- Relative pathnames (start where you are)
- Important standard directories:
  - / /home /usr /usr/bin
  - /sbin /usr/sbin /etc /var
  - /dev /tmp /usr/ucb

The Home Directory

- Your home directory is defined in /etc/passwd
- You can use any of the following commands to get home:
  - cd
  - cd ~
  - cd $HOME
  - cd /full/path/to/your/home/dir
- Change to the previous (last) directory:
  - cd -

Man Pages

- Man man
- Sections:
  - user
  - system calls
  - C library functions/X Windows functions
  - devices and network interfaces
  - file formats
  - games and demos
  - environments/miscellaneous
  - system administration/maintenance
- whatis *keyword*
- man [1,2] write (1 is user command, 2 is system call)
- man -k *keyword* (or apropos *keyword*)

“Everything in Unix is a File”

- A file has a name and an associated inode, which contains detailed information about the file
- What is a directory?
  - A *file* containing a list of other files, some of which may be other directories
  - A file whose contents is a list of name + inode pairs
- The directory structure is an *abstraction* that *presents* files on a file system in a hierarchical manner—but the files’ contents remain *littered* across the physical file system.
- A file is a stream of bytes
- A file is referenced by a user via a filename. A filename can be up to 255 bytes.
- A file is referenced by the system via the inode

inode details

- Every file is associated with a potentially unique inode. In fact, early Unix systems refered to filenames as “links”, that is, names “linked” to an inode.
- The inode contains information about the file and the inode itself, like:
  - File type (regular, link, directory, etc.)
  - Number of Hard Links to the inode
  - Associated file byte stream length in bytes
  - Device ID where the file is located (/dev/hda1)
  - Inode number of this file
  - File owner’s userid and groupid
  - mtime, atime, and ctime
  - permissions (rwx)
- ls -i (ls -1iF)
- The stat command

File Permissions

- Files have three categories of permissions:
  - user (owner)
  - group
  - other (everyone else NOT in one of the above)
- r (4): Read permission (can open the file)
- w(2): Write permission (can modify it)
- x (1): Execute permission (can run it)

Directory Permissions

- Directories have three categories of permissions:
  - user (owner)
  - group
  - other (everyone else NOT in one of the above)
- r (4): Read permission (can ls the filenames)
- w(2): Write permission (can modify the dir)
- x (1): Execute permission (can cd into dir)
- t (sticky bit): individual ownership only

Links

- Hard Links
  - ln origfile linkfile
  - a directory entry with a unique name referencing a particular inode
  - ls -i will list out inodes for files (ls -1iF)
  - Only superuser can hard link to a directory
  - Hard links are only meaningful within a single filesystem, not across mount points
  - A hard link's *inode* is the same number as the linked file's *inode*
- Soft (Symbolic) Links:
  - ln -s origfile linkfile
  - Anyone can create a soft link to a directory
  - A softlink can refer to another file on another filesystem
  - ls -[l]F will reveal softlinks (noted by -> pointer and @ notation)
  - A softlink's *contents* is the name of the file *pointed to*.

Redirection

- Unix has three default file handles (defined in /usr/include/unistd.h):
  - Standard Output (stdio, 1)
  - Standard Error (stderr, 2)
  - Standard Input (stdin, 0)
- By default, standard output is sent to the current process owner's *terminal*
- Redirection causes the standard output of the current process to go to *some other designated file*:
  - ls -la >/tmp/some.other.file
  - cat /tmp/some.other.file

Here Documents (part un)

- A Here Document allows you to enter a full command that will be executed at some later time (soon, Ralph, very soon)

```
cat <<-EOF
> this is line one
> this is line two
> this is line three
EOF

nl .bashrc

nl <<-EOF
> this is line one
> this is line two
> this is line three
EOF
```

Here Documents (part deux)

- I want to contact ftp.oreilly.com, cd to the /pub/examples/nutshell/sys.prog directory, and get the file examples.tar.gz, all in one command.

```
ftp -n ftp.oreilly.com <<-EOF
passive
user anonymous mark@cs.uchicago.edu
binary
hash
cd /pub/examples/nutshell/sys.prog
dir
get examples.tar.gz
quit
EOF
ls -l examples.tar.gz
```

Here Documents (part trois)

- I want to change all the include lines in a bunch of C source files, and change “#include </wrong/stdio.h>” to “#include <stdio.h>”, all in one command (only works with bash version 2.0 or greater).

```
let “x=0”
> while ((x < 10))
> do
> cp orig.c “file${x}.c”
> let “x=x+1”
> done

for file in *.c
> do
> ed $file <<-EOF
> g/\/wrong\//s///
> w
> q
> EOF
> done
```

- Help on ed editor is at http://www.bath.ac.uk/bucs/Docs/ub-4-1.html

Shell Quoting (Maskierungen)

- \ protects the next character from the shell's interpretation except the newline character
- "" protects everything from the shell's interpretation *except* double quotes, backslashes, dollar signs, and backquotes
- '' protects everything from the shell's interpretation except single quotes

Shell Variables

- foo=“hello world”
- echo $foo
- echo “$foo”
- echo “\$foo”
- echo ‘$foo’
- echo ‘\$foo’

Job Control

- & puts current process in background
- jobs prints out current jobs in shell
- kill %*n* terminates a given job
- fg [%*n*] moves a job to the foreground
- bg [%*n*] moves a job to the background

Filters

- Filters are programs that are written to accept input from STDIN in addition to any other forms of input.
- Filters send output to STDOUT.
- The list of Unix Filters includes:
  - cat
  - cut
  - less
  - grep
  - sort
  - tr
  - uniq
  - wc
  - tail, head
  - lpr

cat and sort

cd ~mark/pub/51081

cat sort.txt

sort +1 -2 sort.txt (sorts by last name)

- sort numerically

  sort +2 -3 sort.txt (sort by year of inauguration)

  sort +3 -4 sort.txt (won't work)

  sort +3 -4 -n sort.txt (specify numeric sort)

- sort passwd file by userid

  sort -t: -n +2 -3 passwd2

- practice more on sort3.txt

  sort sort3.txt

  sort -n +1 -2 sort3.txt

sort as a Filter

- Sort the password file by userid numerically

  cat passwd2 | sort -t: -n +2 -3
- list files by decreasing numbers of lines per file

  wc -l * 2>/dev/null |sort -rn
- now, get rid of total line:

  wc -l * 2>/dev/null | sort -rn | tail +2
- other examples:

  ls -la | sort -n +4 -5

  ls -la | sort -n -k5

uniq Filter

- uniq (cd to ~mark/pub/51081 subdirectory)

  uniq uniq.txt (does nothing: uniq only works on adjacent lines)
- how can we get adjacent lines?

  sort uniq.txt

  sort uniq.txt | uniq

  sort -u uniq.txt
- how many duplicates (-c = count)?

  sort uniq.txt | uniq -c
- Print just list the duplicated lines, once (lists only lines duplicated=2 or more appearances)

  sort uniq.txt | uniq -d
- List NON-duplicated lines (lines appearing only once = singletons)

  sort uniq.txt |uniq -u

cut Filter

- cut (filters text from files - cuts columns and fields)
  - column cutting -c
    - who | cut -c10-17
    - who | cut -c19-31
    - who | cut -c32-
  - tail passwd2
  - field cutting -f
    - cut -d: -f1,5 passwd2 | tail
    - cat passwd2 | cut -d: -f1,5 | tail
    - cat passwd2 | cut -d: -f1,5-6 | tail
    - cat passwd2 | cut -d: -f1,5-6 | grep mooney