# Contents





# Standardization and Implementation

- Why Standardization?
  - Proliferation of UNIX versions
- What should be done?
  - The specifications of limits that each implementation must define!

# UNIX Standardization

- ANSI C
  - American National Standards Institute
  - ISO/IEC 9899:1990
    - International Organization for Standardization (ISO)
    - Syntax/Semantics of C, a standard library
  - Purpose:
    - Provide portability of conforming C programs to a wide variety of OS's.
  - 15 areas: Fig 2.1 – Page 27



# UNIX Standardization

- ANSIC C
  - <assert.h> - verify program assertion
  - <ctype.h> - char types
  - <errno.h> - error codes
  - <float.h> - float point constants
  - <limits.h> - implementation constants
  - <locale.h> - locale catalogs
  - <math.h> - mathematical constants
  - <setjmp.h> - nonlocal goto
  - <signal.h> - signals
  - <stdarg.h> - variable argument lists
  - <stddef.h> - standard definitions
  - <stdio.h> - standard library
  - <stdlib.h> - utilities functions
  - <string.h> - string operations
  - <time.h> - time and date

# UNIX Standardization

- POSIX.1 (Portable Operating System Interface) developed by IEEE
  - Not restricted for Unix-like systems and no distinction for system calls and library functions
  - Originally IEEE Standard 1003.1-1988
  - 1003.2: shells and utilities, 1003.7: system administrator, > 15 other communities
  - Published as IEEE std 1003.1-1990, ISO/IEC9945-1:1990
  - New: the inclusion of symbolic links
  - No superuser notion



# UNIX Standardization

- POSIX.1
  - <cpio.h> - cpio archive values
  - <dirent.h> - directory entries
  - <fcntl.h> - file control
  - <grp.h> - group file
  - <pwd.h> - passwd file
  - <tar.h> tar archieve values
  - <termios.h> - terminal I/O
  - <unistd.h> - symbolic constants
  - <utime.h> file times
  - <sys/stat.h> - file status
  - <sys/times.h> - process times
  - <sys/types.h> - primitive system data types
  - <sys/utsname.h> - system name
  - <sys/wait.h> - process control

# UNIX Standardization

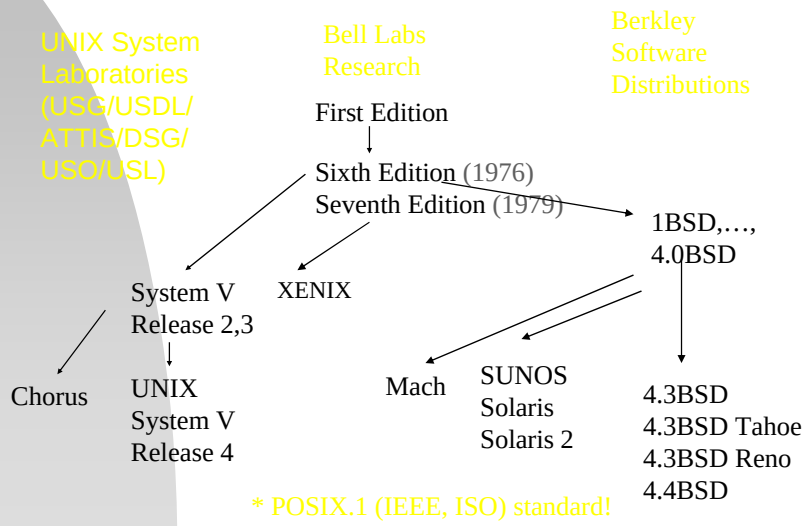
- X/Open
  - An international group of computer vendors
  - Volume 2 of X/Open Portability Guide, Issue 3 (XPG3)
    - XSI System Interface and Headers
    - Based on IEEE Std. 1003.1 – 1988 (text displaying in different languages)
    - Built on the draft of ANSI C
      - Some are out-of-date.
  - Solaris 2.4 – compliance to XPG4V2
    - man xpg4



# UNIX Standardization

- FIPS (Federal Information Processing Standard) 151-1
  - IEEE Std. 1003.1-1988 & ANSI C
  - For the procurement of computers by the US government.
  - Required Features:
    - JOB_CONTROL, SAVED_ID, NO_TRUNC, CHOWN_RESTRICTED, VDISIBLE,
    - NGROUP_MAX >= 8, Group Ids of new files and dir be equal to their parent dir, env var HOME and LOGNAME defined for a login shell, interrupted read/write functions return the number of transferred bytes.

# UNIX Implementation






# UNIX Implementation

- System V Release 4 - 1989
  - POSIX 1003.1 & X/Open XPG3
  - Merging of SVR3.2, SunOS, 4.3BSD, Xenix
  - SVID (System V Interface Definition)
    - Issue 3 specifies the functionality qualified for SVR4.
  - Containing of a Berkley compatibility library
    - For 4.3BSD counterparts

# UNIX Implementation

- 4.2BSD - 1983
  - DARPA (Defense Advanced Research Projects Agency) wanted a standard research operating systems for the VAX.
  - Networking support - remote login, file transfer (ftp), etc. Support for a wide range of hardware devices, e.g., 10Mbps Ethernet.
  - Higher-speed file system.
  - Revised virtual memory to support processes with large sparse address space (not part of the release).
  - Inter-process-communication facilities.



# UNIX Implementation

- 4.3 BSD - 1986
  - Improvement of 4.2 BSD
    - Loss of performance because of many new facilities in 4.2 BSD.
    - Bug fixing, e.g., TCP/IP implementation.
    - New facilities such as TCP/IP subnet and routing support.
  - Backward compatibility with 4.2 BSD.
  - Second Version - 4.3 BSD Tahoe
    - support machines beside VAX
  - Third Version - 4.3 BSD Reno
    - freely redistributable implementation of NFS, etc.

# UNIX Implementation

- 4.4 BSD - 1992
  - POSIX compatibility
  - Deficiencies remedy of 4.3 BSD
    - Support for numerous architectures such as 68K, SPARC, MIPS, PC.
    - New virtual memory better for large memory and less dependent on VAX architecture – Mach.
    - TCP/IP performance improvement and implementation of new network protocols.
    - Support of an object-oriented interface for numerous filesystem types, e.g., SUN NFS.



# UNIX Implementation - Major UCB CSRG Distributions

- Major new facilities:
  - 3BSD, 4.0BSD, 4.2BSD, 4.4 BSD
- Bug fixes and efficiency improvement:
  - 4.1 BSD, 4.3BSD
- BSD Networking Software, Release 1.0 (from 4.3BSD Tahoe, 1989), 2.0 (from 4.3BSD Reno, 1991)
- Remark:
  - Standards define a subset of any actual system – compliance and compatibility

# Limits – ANSI C, POSIX, XPG3, FIPS 151-1

- Compiler-time options and limits (headers)
  - Job control?
  - Largest value of a short?
- Run-time limits related to file/dir
  - pathconf and fpathconf, e.g., the max # of bytes in a filename
- Run-time limits not related to file/dir
  - sysconf, e.g., the max # of opened files per process
- Remark: implementation-related



# ANSI C Limits

- All compile-time limits - <limits.h>
  - Minimum acceptable values
    - E.g., CHAR_BIT, INT_MAX
  - Implementation-related
    - char (limits.h), float (FLT_MAX in float.h), open (FOPEN_MAX & TMP_MAX in stdio.h)

```
#if defined(_CHAR_IS_SIGNED)
#define CHAR_MAX        SCHAR_MAX
#elif defined(_CHAR_IS_UNSIGNED)
#define CHAR_MAX        UCHAR_MAX
```

# POSIX Limits

- 33 limits and constants
  - Invariant minimum values (POSIX defined in Figure 2.3 – Page 33, limits.h)
  - Corresponding implementation (limits.h)
    - Invariant SSIZE_MAX
    - Run-time increasable value NGROUP_MAX
    - Run-time invariant values, e.g., CHILD_MAX
    - Pathname variable values, e.g., LINK_MAX
  - Compile-time symbolic constants, e.g., _POSIX_JOB_CONTROL
  - Execution-time symbolic constants, e.g., _POSIX_CHOWN_RESTRICTED
  - Obsolete constant: CLK_TCK



# POSIX Limits

- Limitation of POSIX
  - E.g., _POSIX_OPEN_MAX in <limits.h>
  - sysconf(), pathconf(), fpathconf() at run-time
  - Possibly indeterminate from some
    - E.g., OPEN_MAX under SVR4

# XPG3 Limits

- 7 constants in <limits.h> - invariant minimum values called by POSIX.1
  - Dealing with message catalogs
    - NL_MSGMAX – 32767
- PASS_MAX
  - <limits.h>
    - Run-time invariant value called by POSIX.1
  - sysconf()



# Run-Time Limits

- #include <unistd.h> (Figure 2.7 – Page 40: compile/run time limits)
- long sysconf(int *name*);
  - _SC_CHILD_MAX, _SC_OPEN_MAX, etc.
- long pathconf(const char **pathname*, int *name*);
- long fpathconf(int **filedes*, int *name*);
  - _PC_LINK_MAX, _PC_PATH_MAX, _PC_PIPE_BUF, _PC_NAME_MAX, etc.
- Various *names* and restrictions on arguments (Page 35 and Figure 2.5)
- Return –1 and set errno if any error occurs.
  - EINVAL if the name is incorrect.

# Run-Time Limits

- Example Program 2.1 – Page 38
  - Print sysconf and pathconf valuesb (Fig 2.6 – Page 39)

| Limit | SunOS4.1.1 | SVR4 | 4.3+BSD |
|---|---|---|---|
| CHILD_MAX | 133 | 30 | 40 |
| OPEN_MAX | 64 | 64 | 64 |
| LINK_MAX | 32767 | 1000 | 32767 |
| NAME_MAX | 255 | 14/255 | 255 |
| _POSIX_NO_TRUC | 1 | nodef/1 | 1 |



# Indeterminate Run-Time Limits – Two Cases

- Pathname
  - 4.3BSD: MAXPATHLEN in <sys/param.h>, PATH_MAX in <limits.h>
  - Program 2.2 – Page 42
    - Allocate space for a pathname vs getcwd
    - _PC_PATH_MAX is for relative pathnames
- Max # of Open Files – POSIX run-time invariant
  - NOFILE (<sys/param.h>), _NFILE (stdio.h>)
  - sysconf(_SC_OPEN_MAX) – POSIX.1
    - getrlimit() & setrlimit() for SVR4 & 4.3+BSD
  - Program 2.3 – Page 43: OPEN_MAX!

# MISC

- Feature Test Macro
  - POSIX only
    - cc –D_POSIX_SOURCE file.c
    - Or, #define _POSIX_SOURCE 1
  - ANSI C only

```
ifdef __STDC__
void *myfunc(const char*, int)
#else
void *myfunc();
#endif
```



# MISC

- Primitive System Data Types
  - Figure 2.8 – Page 45
    - Implementation-dependent data types
    - E.g., caddr_t, pid_t, ssize_t, off_t, etc.
  - <sys/types.h>
    - E.g., major_t, minor_t, caddr_t, etc.
    - Examples:
      - typedef char * caddr_t;
      - typedef ulong_t major_t;

      (SRV4: 14 bits for major, 18 bits for minor
      traditionally they are all short: 8-bits)

# MISC

- Conflicts Between Standards
  - clock() in ANSI C and times() in POSIX.1
    - clock_t divided by CLOCKS_PER_SEC in <time.h> (while CLK_TCK became obsolete)
  - Implementation of POSIX functions
    - No assumption on the host operating system.
    - signal() in SRV4 is different from sigaction() in POSIX.1