Advanced Programming in the UNIX Environment

Department of Computer Science
Stevens Institute of Technology

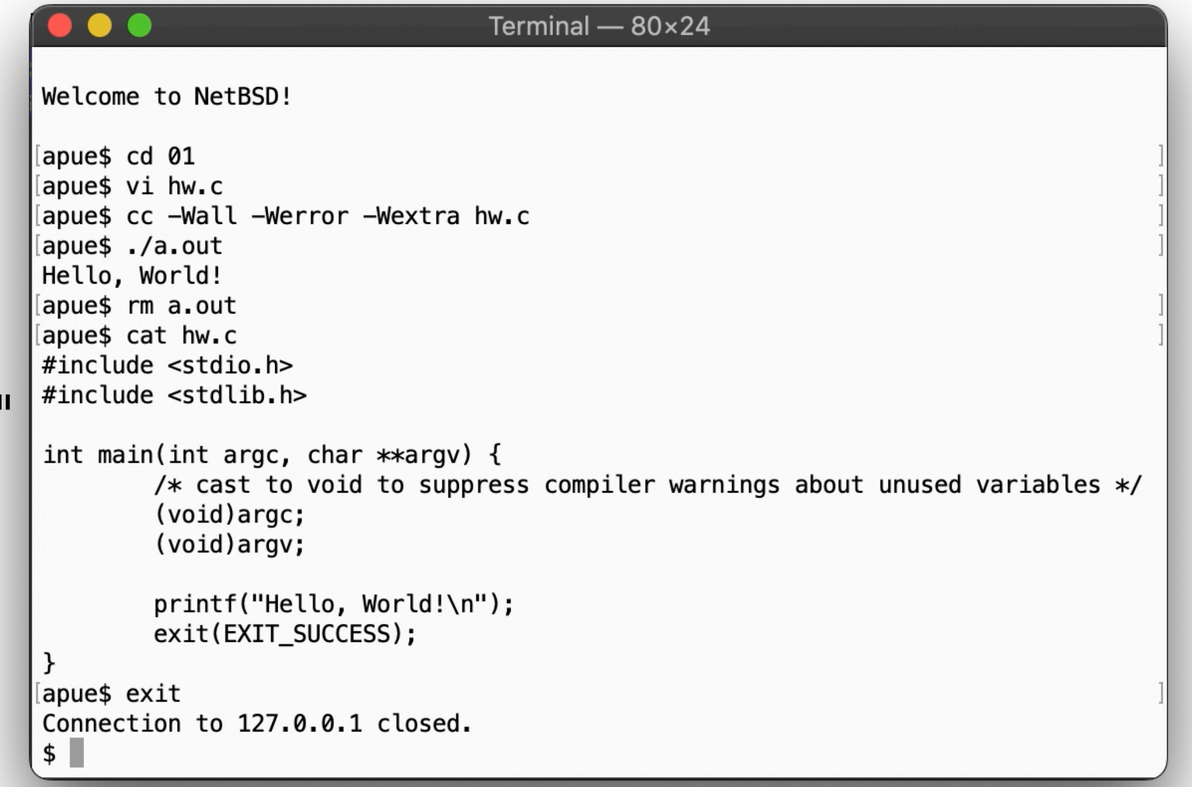
Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

About this class

This class is called "*Advanced Programming in the UNIX Environment*".

It is *not* called:

- "An Introduction to Unix"
- "What Even Is A Programming?"
- "Teach Yourself C Programming in 24 Hours!"



This class in a nutshell: the "what"

Terminal — 80×24

```
apue$ ls /bin
[          cpio       echo       ln         ps         rump.dd    test
cat        csh        ed         ls         pwd        sh
chgrp      date       expr       mkdir      rcmd       sleep
chio       dd         hostname   mt         rcp        stty
chmod      df         kill       mv         rm         sync
cp         domainname ksh        pax        rmdir      tar
apue$ whatis sh
sh(1) - command interpreter (shell)
apue$ whatis ls
ls(1) - list directory contents
apue$ whatis cat
cat(1) - concatenate and print files
apue$ whatis love
love: not found
apue$ whatis whatis
whatis(1) - describe what a command is
apue$
```

This class in a nutshell: the "what"

Terminal — 80×24

```
int     accept(int, struct sockaddr * __restrict, socklen_t * __restrict);
int     accept4(int, struct sockaddr * __restrict, socklen_t * __restrict, int);
int     bind(int, const struct sockaddr *, socklen_t);
int     connect(int, const struct sockaddr *, socklen_t);
int     getpeername(int, struct sockaddr * __restrict, socklen_t * __restrict);
int     getsockname(int, struct sockaddr * __restrict, socklen_t * __restrict);
int     getsockopt(int, int, int, void *__restrict, socklen_t * __restrict);
int     getsockopt2(int, int, int, void *__restrict, socklen_t * __restrict);
int     listen(int, int);
int     paccept(int, struct sockaddr * __restrict, socklen_t * __restrict,
ssize_t recv(int, void *, size_t, int);
ssize_t recvfrom(int, void *__restrict, size_t, int,
ssize_t recvmsg(int, struct msghdr *, int);
ssize_t send(int, const void *, size_t, int);
ssize_t sendto(int, const void *,
ssize_t sendmsg(int, const struct msghdr *, int);
int     setsockopt(int, int, int, const void *, socklen_t);
int     shutdown(int, int);
int     sockatmark(int);
int     socket(int, int, int)
int     socketpair(int, int, int, int *);
int     sendmmsg(int, struct mmsghdr *, unsigned int, unsigned int);
int     recvmmsg(int, struct mmsghdr *, unsigned int, unsigned int,
apue$ grep "(int" /usr/include/sys/socket.h
```

This class in a nutshell: the "what"

- gain an understanding of Unix operating systems
- gain (systems) programming experience
- understand fundamental OS concepts (with focus on Unix family):
  - multi-user concepts
  - basic and advanced I/O
  - process relationships
  - interprocess communications
  - basic network programming using a client/server model

This class in a nutshell: the "why"

- understanding how Unix works gives you insights in other OS concepts
- system level programming experience is invaluable as it forms the basis for most other programming and even *use* of the system
- system level programming in C helps you understand general programming concepts
- most higher level programming languages (eventually) call (or implement themselves) standard C library functions

This class in a nutshell: the "how"

Our reference platform is NetBSD >= 10.0.

You may choose to develop on e.g., your laptop running another OS, but you must make sure that your code compiles and runs flawlessly on NetBSD >= 10.0 using the system provided compiler (gcc >= 10.5.0).

Instructions for how to install NetBSD 10.0 in a VirtualBox VM can be found here:

https://stevens.netmeister.org/631/virtualbox/

Instructions for how to install NetBSD 10.0 in a UTM VM on Apple M1 hardware can be found here:

https://stevens.netmeister.org/631/utm/

This class in a nutshell: the "how"

- https://stevens.netmeister.org/631/#source-code

Terminal — 80×24

```
apue$ cat fetch-sources.sh
#! /bin/sh
set -eu

umask 022

for set in gnusrc sharesrc src syssrc; do
        echo "Fetching ${set}..."
        ftp -V ftp.netbsd.org:/pub/NetBSD/NetBSD-9.0/source/sets/${set}.tgz
        echo "Extracting ${set}..."
        su root -c "tar zxf ${set}.tgz -C /"
        rm -f "${set}.tgz"
done
apue$ ls -l /usr/src
ls: /usr/src: No such file or directory
apue$ sh fetch-sources.sh
Fetching gnusrc...
Extracting gnusrc...
Fetching sharesrc...
Extracting sharesrc...
Fetching src...
Extracting src...
Fetching syssrc...
Extracting syssrc...
```

Terminal — 80×24

```
/* fts_build flags */
#define BCHILD          1               /* fts_children */
#define BNAMES          2               /* fts_children, names only */
#define BREAD           3               /* fts_read */

#ifndef DTF_HIDEW
#undef FTS_WHITEOUT
#endif

FTS *
fts_open(char * const *argv, int options,
    int (*compar)(const FTSENT **, const FTSENT **))
{
        FTS *sp;
        FTSENT *p, *root;
        size_t nitems;
        FTSENT *parent, *tmp = NULL;    /* pacify gcc */
        size_t len;

        _DIAGASSERT(argv != NULL);

        /* Options check. */
"/usr/src/lib/libc/gen/fts.c" [readonly] 1244L, 32234C        118,1          8%
```

- https://stevens.netmeister.org/631/compare-code-exercise.html

This class in a nutshell: the "how"

We will write a fair bit of code in this class.

Writing code is communication.

Make sure your code is:

- clearly structured
- well-formatted
- uses a consistent coding style (indentation, placement of braces, etc.)
- variables, functions etc. are sensibly named
- comments are used only when necessary, explaining the *why*, not the *how*

See also: https://stevens.netmeister.org/631/style

About this class

Textbook:

"Advanced Programming in the UNIX® Environment", Third Edition, by W. Richard Stevens, Stephen A. Rago

Grading:

- course participation, checkpoints: 50 points
- 2 smaller homework assignments, worth 25 points each
- 1 midterm project, worth 100 points
- 1 final project (group work), worth 200 points
- 1 final programming assignment (individual), worth 100 points

About this class

You are responsible for your work as well as your time management. If you run into challenges, contact me as soon as possible and we will work something out.

There will be no extra-credit assignments, but you may resubmit your work to address any problems identified to improve your grade.

You are responsible for your own work. You may not present as your own the ideas, code, or code samples of another, even if those are available on the internet. Any incidents of plagiarism and copyright infringement will be reported to the Dean of Graduate Academics.

https://stevens.netmeister.org/631/#cheating

Permitted use of (generative) AI technologies

You *may* use AI programs such as e.g., ChatGPT to help generate ideas and brainstorm.

Note that the material generated by these programs may be inaccurate, incomplete, or otherwise problematic and often stifles your own independent thinking and creativity.

You *may not* submit any work generated by an AI program as your own. If you include material generated by an AI program, it should be cited like any other reference material and must include the prompt you used to have the AI to generate the code in question.

Treat these programs like a virtual fellow student: you are allowed to "discuss" with them at a *conceptual* level, but you cannot take their code and hand it in as your own, even if you make minor changes yourself afterwards.

https://stevens.netmeister.org/631/use-of-ai.html

Syllabus

- Introduction, UNIX history, UNIX Programming Basics
- File I/O, File Sharing
- Files and Directories
- Filesystems, System Data Files, Time & Date
- UNIX tools: make(1), gdb(1), revision control, etc.
- Process Environment, Process Control
- Process Groups, Sessions, Signals
- Interprocess Communication
- Daemon Processes, shared libraries
- Advanced I/O: Nonblocking I/O, Polling, and Record Locking
- Encryption
- Code reading, coding style, best practices
- Review

Course Resources

Course Website: https://stevens.netmeister.org/631/

Course Mailinglist: https://lists.stevens.edu/mailman/listinfo/cs631apue

Course Slack: https://cs631apue2024.slack.com/

Course Videos: https://youtube.com/c/cs631apue

Homework

Before *every* lecture:

- review the previous week's materials
- watch the video lectures and slides for that class
- run all examples from the video / slides
- follow up with questions on the course mailing list
- prepare for class by reading the assigned chapters
- do the recommended exercises
- submit the weekly checkpoint

After *every* lecture:

- re-run all examples from the video / slides
- review your notes from class

Homework

Week 1:

- bookmark the course resources
- double-check that you are subscribed to the class mailing list
- customize your NetBSD VM for development
- join the course Slack channel and participate