

# **Advanced Programming in the UNIX Environment**

## **Week 01, Segment 3: UNIX Basics**

**Department of Computer Science  
Stevens Institute of Technology**

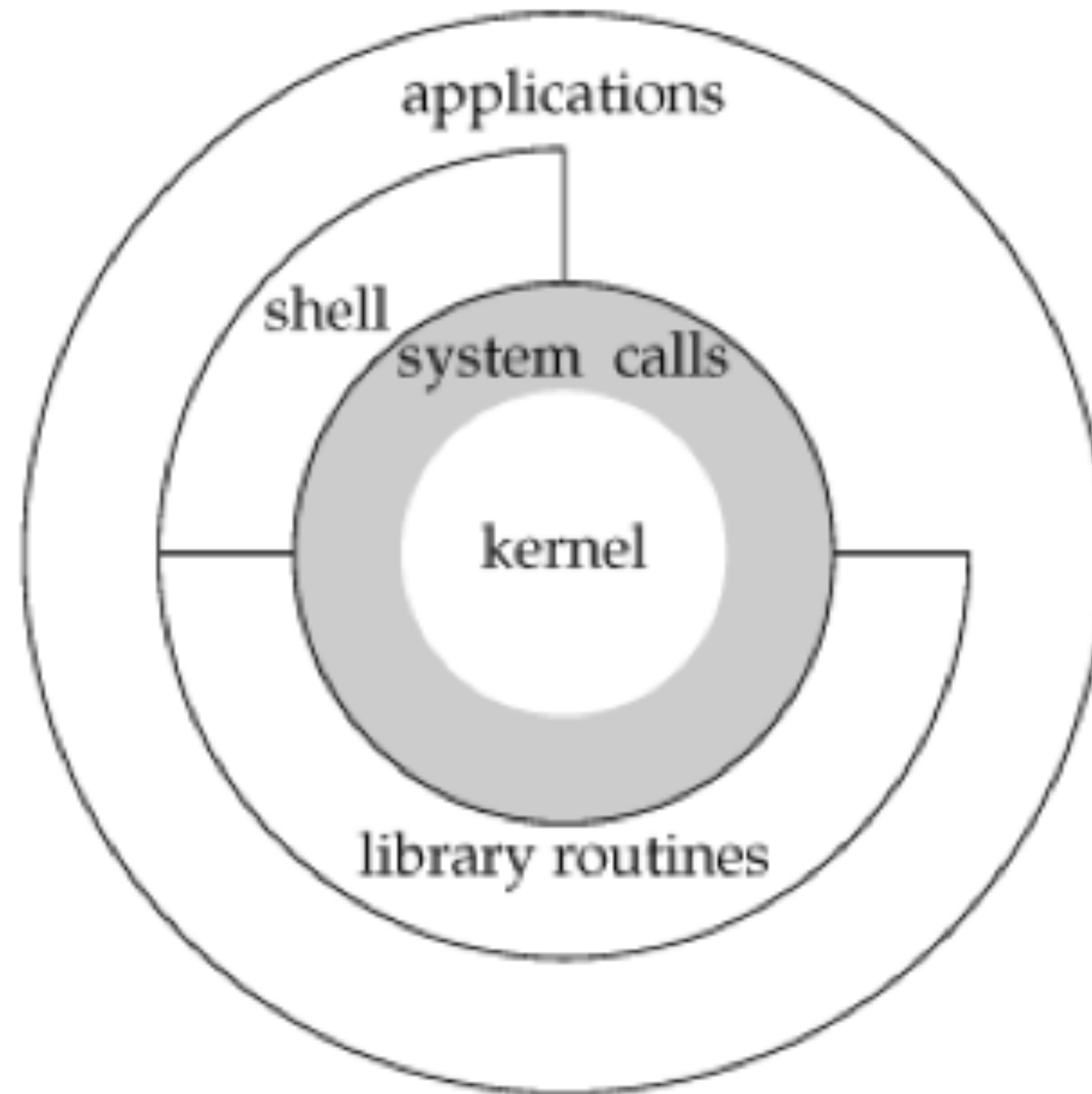
**Jan Schaumann**

`jschauma@stevens.edu`

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

## UNIX Basics: OS Design

---



# System Calls and Library Functions, Standards

---

## System Calls and Library Functions

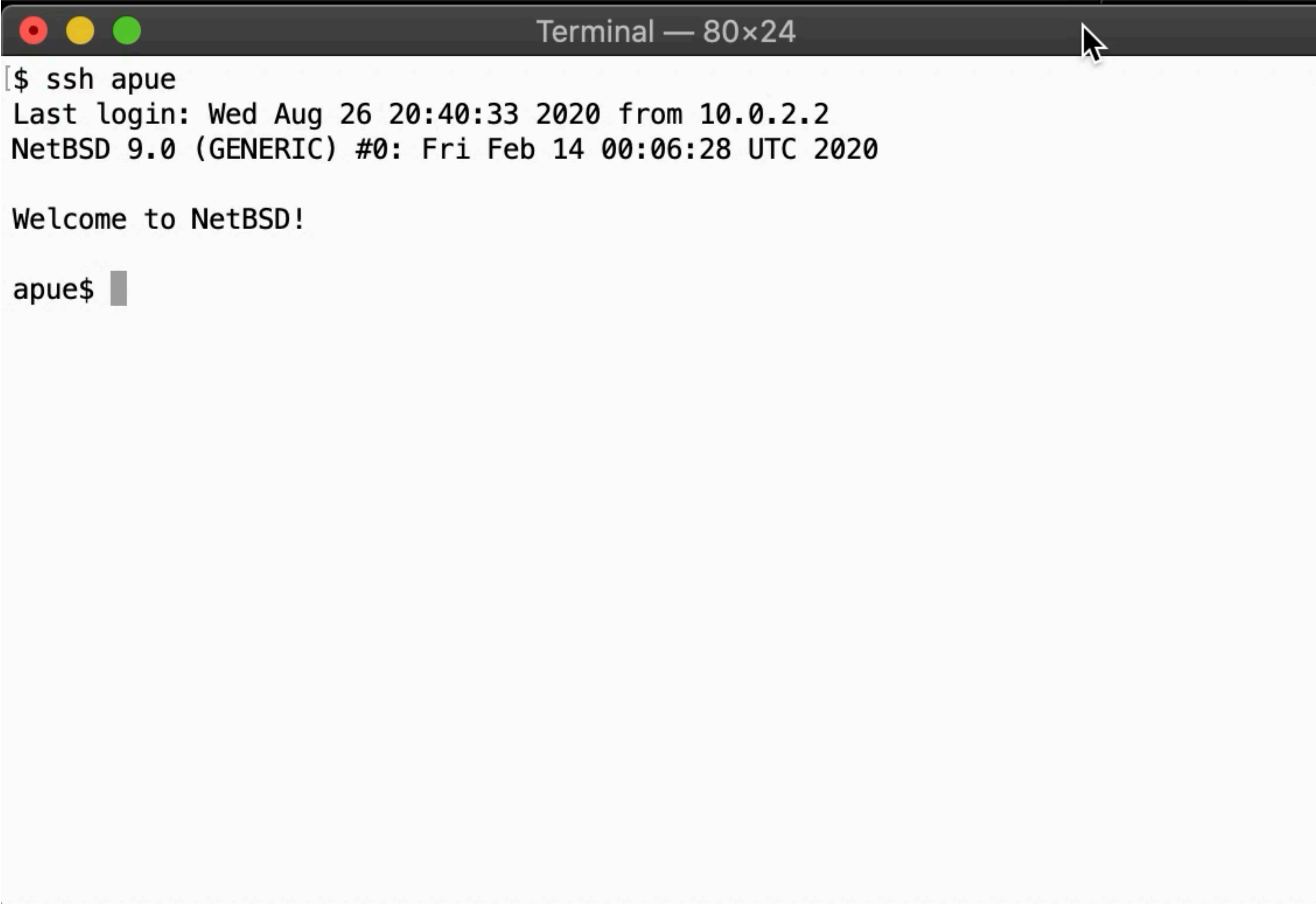
- *System calls* are entry points into kernel code where their functions are implemented. They are documented in section 2 of the manual pages (e.g. `write(2)`).
- *Library calls* are transfers to user code which performs the desired functions. They are documented in section 3 of the manual pages (e.g. `printf(3)`).

## Standards

- IEEE POSIX (1003.1-2008) / SUSv4

# System Calls and Library Functions, Standards

---



```
Terminal — 80x24
[$ ssh apue
Last login: Wed Aug 26 20:40:33 2020 from 10.0.2.2
NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020

Welcome to NetBSD!

apue$
```



INDEX

Search..

[\[Alphabetic\]](#) | [Topic](#) | [Word Search](#)


Select a Volume:  
[\[Base Definitions\]](#) | [System Interfaces](#) | [Shell & Utilities](#) | [Rationale](#)


[\[Frontmatter\]](#)

[\[Main Index\]](#)

Base Definitions

- [1. Introduction](#)
- [2. Conformance](#)
- [3. Definitions](#)
- [4. General Concepts](#)
- [5. File Format Notation](#)
- [6. Character Set](#)
- [7. Locale](#)
- [8. Environment Variables](#)
- [9. Regular Expressions](#)
- [10. Directory Structure and Devices](#)
- [11. General Terminal Interface](#)
- [12. Utility Conventions](#)
- [13. Headers](#)

The Open Group Base Specifications Issue 7,  
2018 edition  
IEEE Std 1003.1™-2017 (Revision of IEEE Std 1003.1-2008)  
[Copyright](#) © 2001-2018 IEEE and The Open Group



POSIX.1-2017 is simultaneously IEEE Std 1003.1™-2017 and The Open Group Technical Standard Base Specifications, Issue 7.

POSIX.1-2017 defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level. POSIX.1-2017 is intended to be used by both application developers and system implementors and comprises four major components (each in an associated volume):

- General terms, concepts, and interfaces common to all volumes of this standard, including utility conventions and C-language header definitions, are included in the Base Definitions volume.
- Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume.
- Definitions for a standard source code-level interface to command interpretation services (a “shell”) and common utility programs for application programs are included in the Shell and Utilities volume.
- Extended rationale that did not fit well into the rest of the document structure, which contains historical information concerning the contents of POSIX.1-2017 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume.

The following areas are outside the scope of POSIX.1-2017:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2017 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

**Keywords**  
application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch



# System Calls and Library Functions, Standards

---

## The C Standard:

- ANSI C (X3.159-1989), aka C89, aka C90; C99 (ISO/IEC 9899); C11 (ISO/IEC 9899:2011); C18 (ISO/IEC 9899:2018); coming soon(?): C2X

## Important ANSI C features:

- function prototypes
- generic pointers (void \*)
- abstract data types (e.g. pid\_t, size\_t)

## Error handling:

- meaningful return values
- errno variable
- lookup of constant error values via two convenient functions:  
strerror(3) and perror(3)

## Let's write some code already!

---

```
$ ftp https://stevens.netmeister.org/631/apue-code.tar.gz
```

```
$ tar zxvf apue-code.tar.gz
```

```
$ cd 01
```

```
$ cc welcome.c
```

```
$ ./a.out
```

```
Welcome to CS631 Advanced Programming in the UNIX Environment, jschauma!
```

```
$
```

## Let's write some code already!

---





## Let's write some code already!

---

```
$ echo "CFLAGS='-Wall -Werror -Wextra'" >> ~/.shrc
```

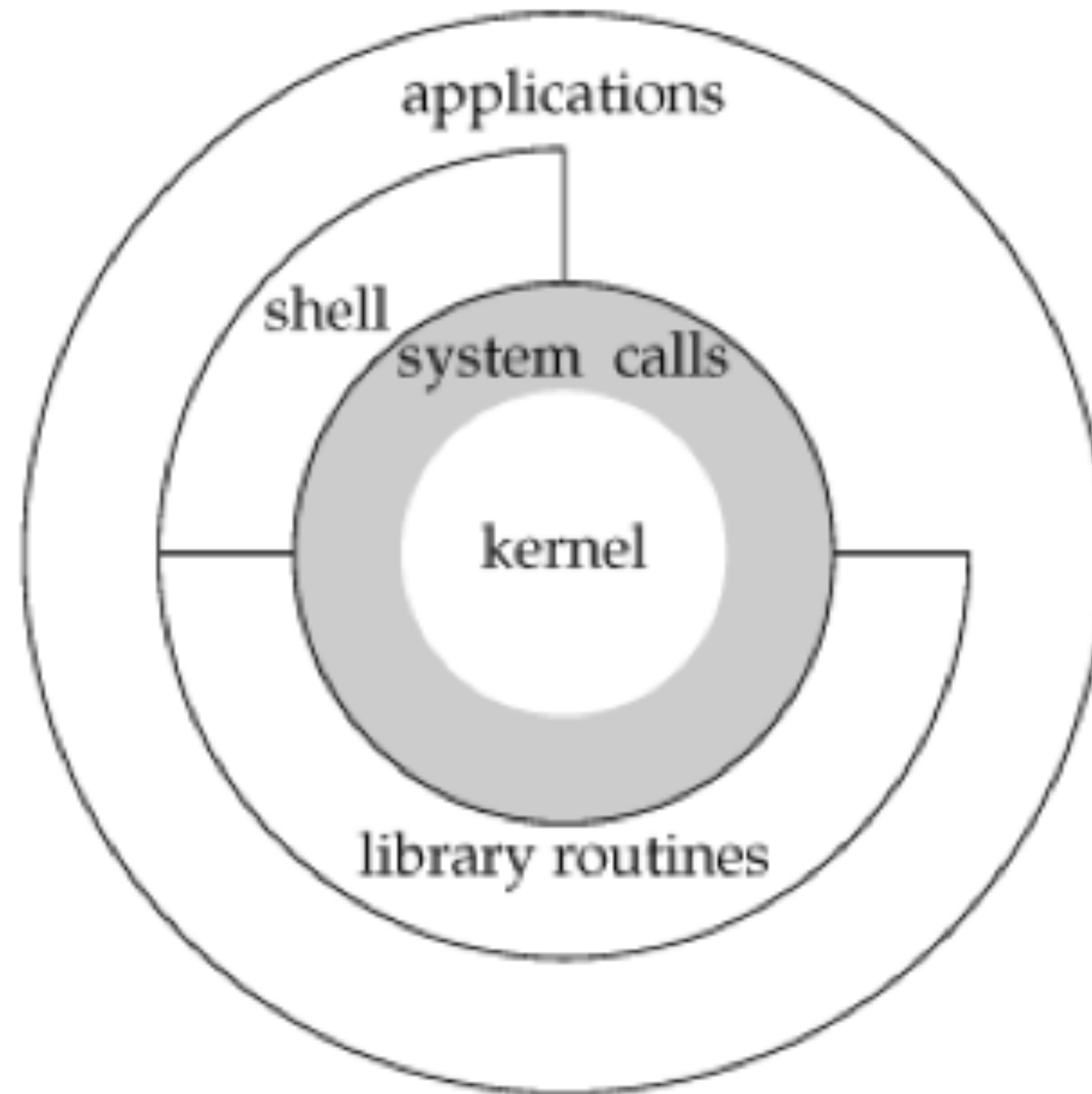
```
$ echo "alias cc='cc \${CFLAGS}'" >> ~/.shrc
```

See also:

<https://kristerw.blogspot.com/2017/09/useful-gcc-warning-options-not-enabled.html>

## UNIX Basics: OS Design

---



## What exactly is a shell?

---

```
$ cd 01
```

```
$ more simple-shell.c
```

```
$ cc -Wall -Werror -Wextra simple-shell.c
```

```
$ ./a.out
```

```
$$ ls
```

```
$$ ls -l
```

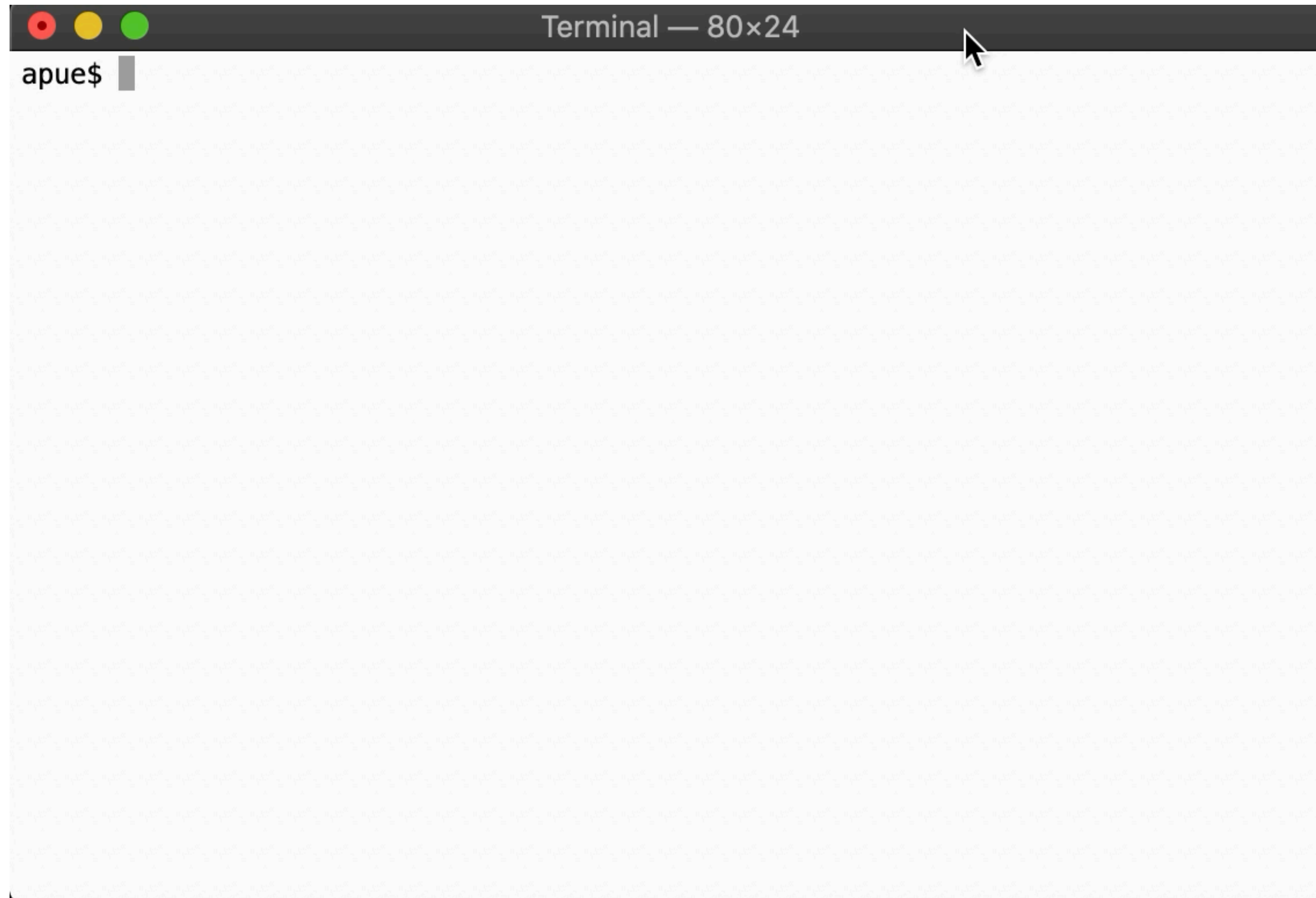
```
$$ exit
```

```
$$ ^D
```

```
$
```

# What exactly is a shell?

---





## Program Design

---

”Consistency underlies all principles of quality.”  
- Frederick P. Brooks, Jr

## Program Design

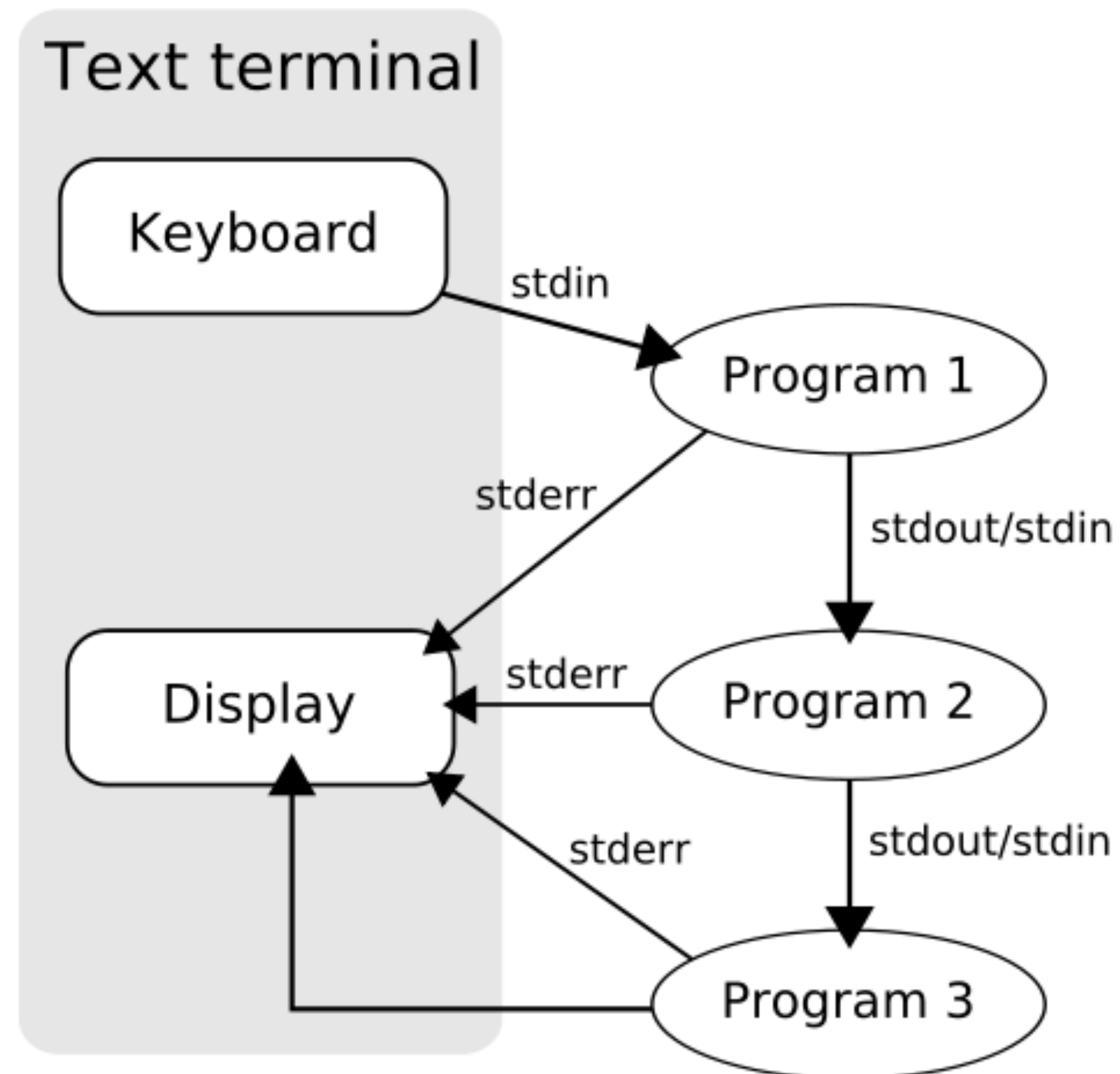
---

[https://en.wikipedia.org/wiki/Unix\\_philosophy](https://en.wikipedia.org/wiki/Unix_philosophy)

UNIX programs...

- ...are simple
- ...follow the element of least surprise
- ...accept input from stdin
- ...generate output to stdout
- ...generate meaningful error messages to stderr
- ...have meaningful exit codes
- ...have a manual page

## UNIX Basics: Pipes



## UNIX Basics: Pipes

---

What is the longest word found on the ten most frequently retrieved English Wikipedia pages?

```
$ for f in $(curl -L https://is.gd/c6F2fs | zgrep -i "^en " | sort -k3 -n | tail -10 |
    sed -e 's/en \(.*\) [0-9]* [0-9]* /\1/'); do
    links -dump https://en.wikipedia.org/wiki/${f};
done |
tr '[:punct:]' ' ' |
tr '[:space:]' '\n' |
tr '[:upper:]' '[:lower:]' | egrep '^[a-z]+$' |
awk '{ print length() " " $0; }' | sort |
uniq |
sort -n |
tail -1
```



## Files and Directories

---

The UNIX filesystem is a tree structure, with all partitions mounted under the root (/). File names may consist of any character except / and NUL as pathnames are a sequence of zero or more filenames separated by /'s.

Directories are special "files" that contain mappings between inodes and filenames, called directory entries.

All processes have a current working directory from which all relative paths are specified. (Absolute paths begin with a slash, relative paths do not.)

## Listing files in a directory

---

```
$ cd 01
```

```
$ more simple-ls.c
```

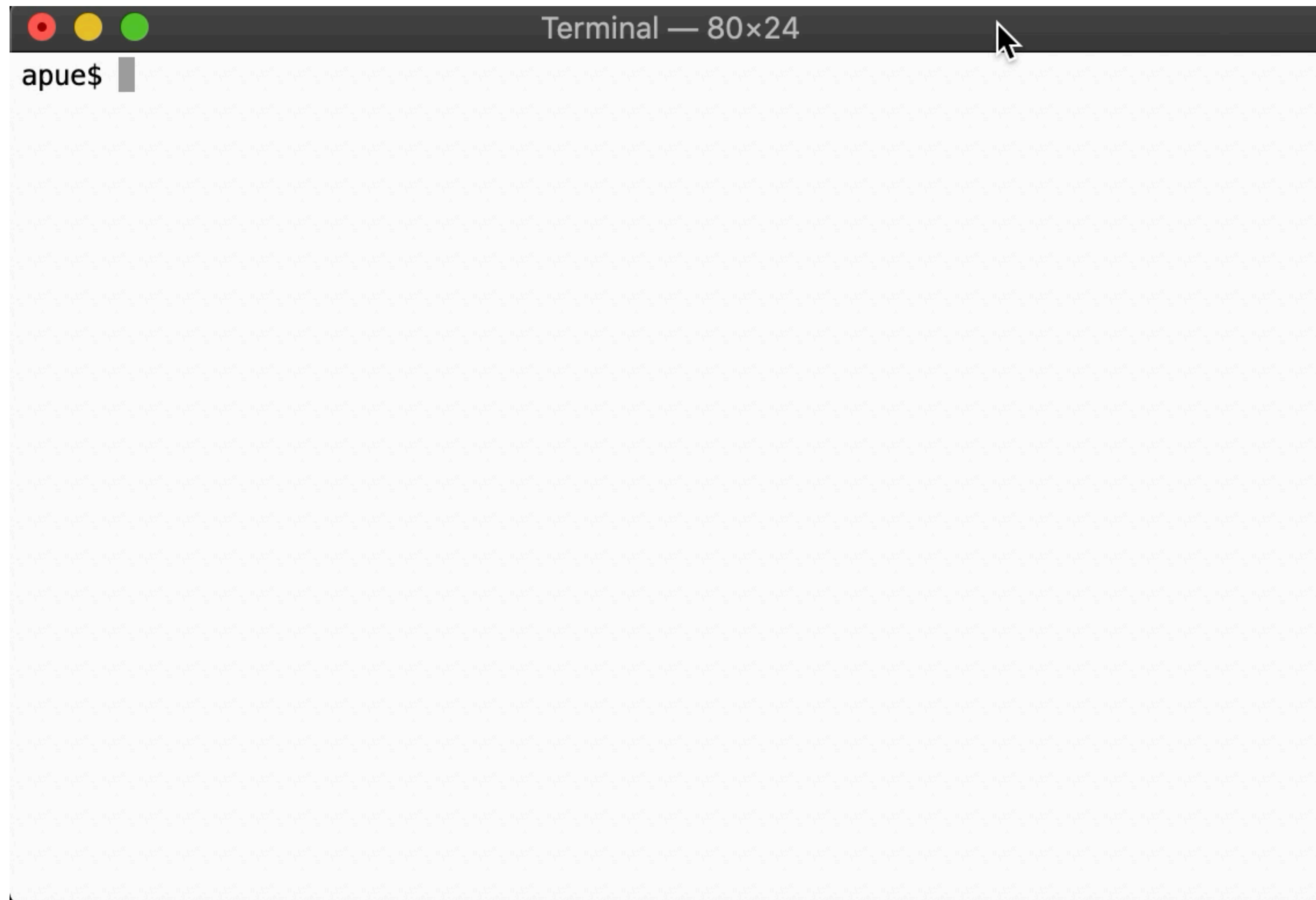
```
$ cc -Wall -Werror -Wextra simple-ls.c
```

```
$ ./a.out .
```

```
$
```

## Listing files in a directory

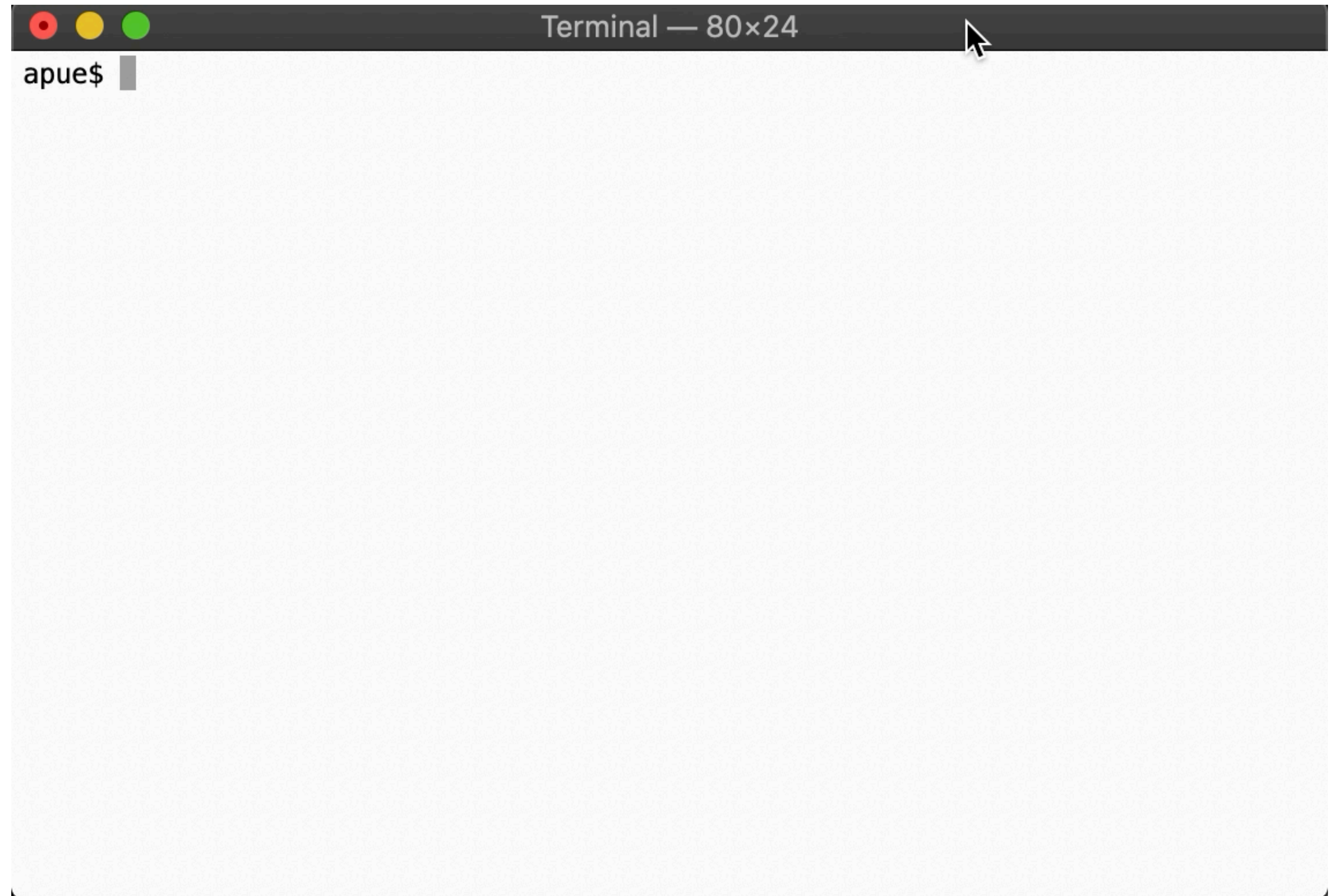
---



## User Identification

User IDs and group IDs are numeric values used to identify users on the system and grant permissions appropriate to them.

Group IDs come in two types; primary and secondary.





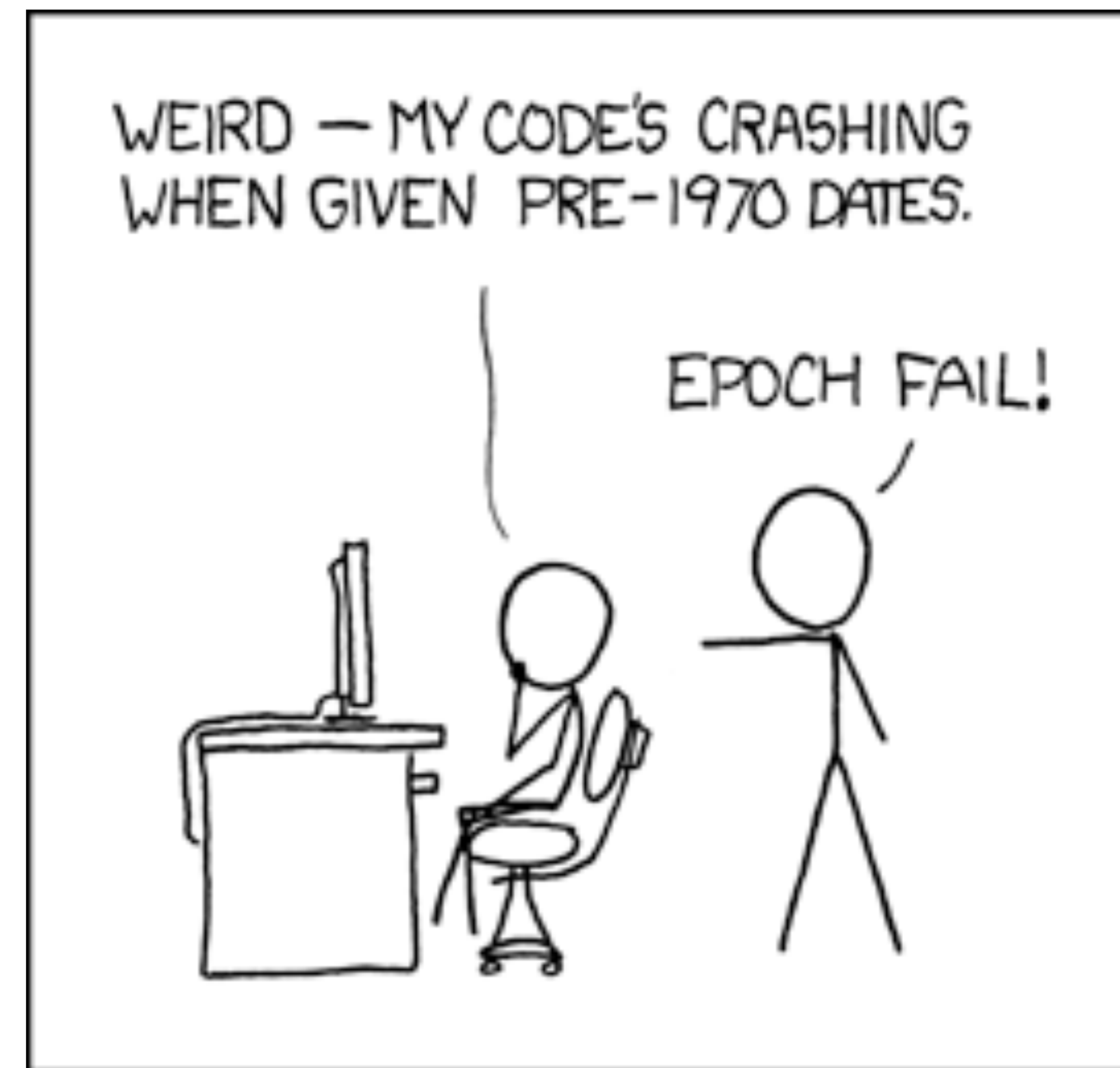
## Unix Time Values

---

Calendar time: measured in seconds since the UNIX epoch (Jan 1, 00:00:00, 1970, GMT). Stored in a variable of type `time_t`.

```
$ date +%s
```

```
1598535631
```



<https://www.xkcd.com/376/>

## Unix Time Values: Year 2038 Problem

---

Binary : 01111111 11111111 11111111 11110000

Decimal : 2147483632

Date : 2038-01-19 03:13:52 (UTC)

Date : 2038-01-19 03:13:52 (UTC)

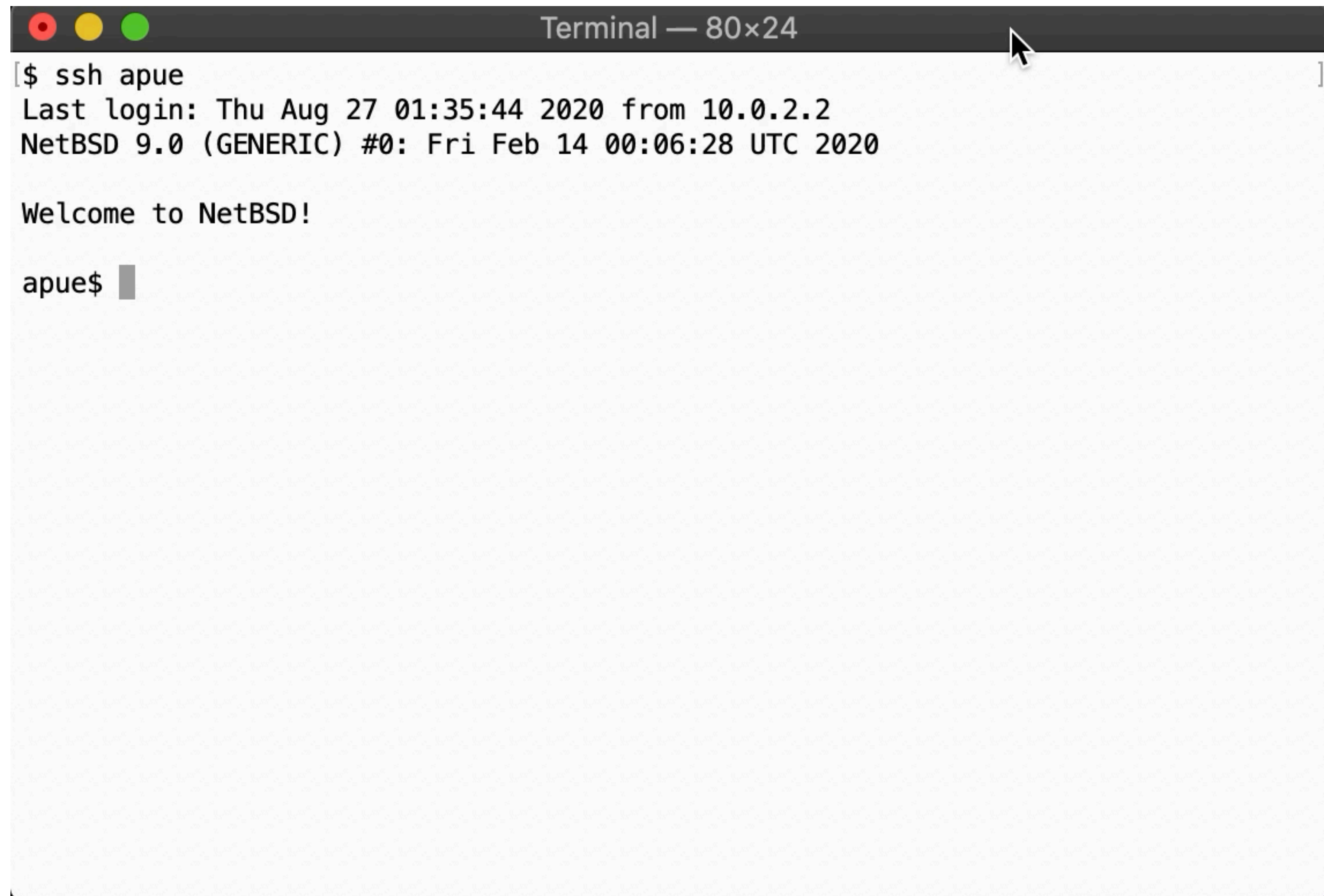
[https://en.wikipedia.org/wiki/Year\\_2038\\_problem](https://en.wikipedia.org/wiki/Year_2038_problem)

## Unix Time Values

*Process time*: central processor resources used by a process.

Measured in clock ticks (clock\_t). Three values:

- clock time
- user CPU time
- system CPU time

A terminal window titled "Terminal — 80x24" with standard macOS window controls (red, yellow, green buttons). The terminal shows an SSH session initiated from a local machine to a remote machine named "apue". The output includes the last login time, the NetBSD version (9.0), and a welcome message. The prompt "apue\$" is visible at the bottom.

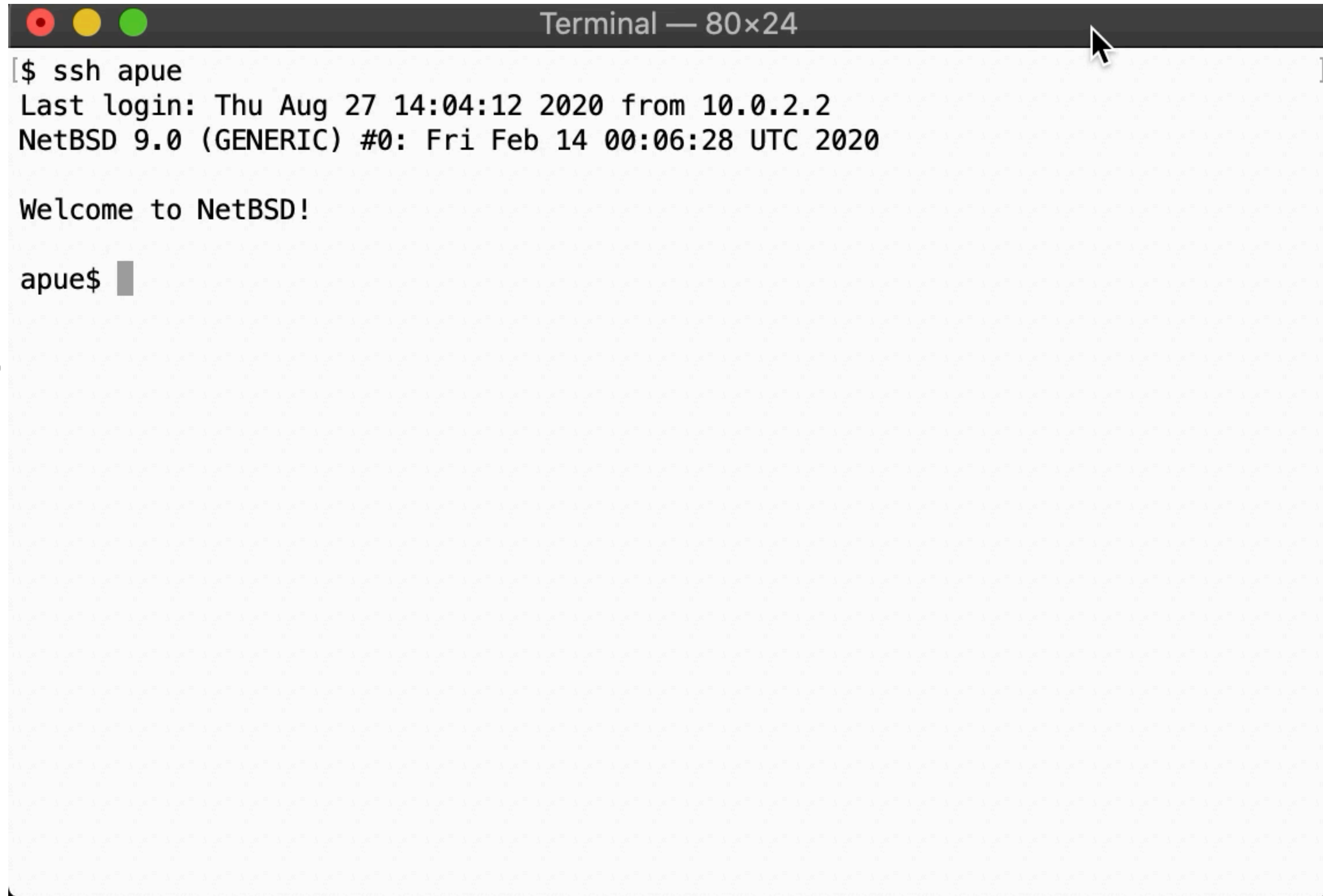
```
[ $ ssh apue
Last login: Thu Aug 27 01:35:44 2020 from 10.0.2.2
NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020

Welcome to NetBSD!

apue$
```

## Standard I/O

- **file descriptors**: Small, non-negative integers which identify a file to the kernel. The shell can redirect any file descriptor.
- kernel provides **unbuffered** I/O through e.g. `open(2)`, `read(2)`, `write(2)`, `lseek(2)`, `close(2)`
- kernel provides **buffered** I/O through e.g. `fopen(3)`, `fread(3)`, `fwrite(3)`, `getc(3)`, `putc(3)`

A terminal window titled "Terminal — 80x24" with a dark gray title bar and three colored window control buttons (red, yellow, green) on the left. The terminal content shows an SSH session: the prompt is "\$", the command is "ssh apue", and the output shows the last login time, the NetBSD version (9.0), and a welcome message. The prompt then changes to "apue\$".

```
Terminal — 80x24
[$ ssh apue
Last login: Thu Aug 27 14:04:12 2020 from 10.0.2.2
NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020

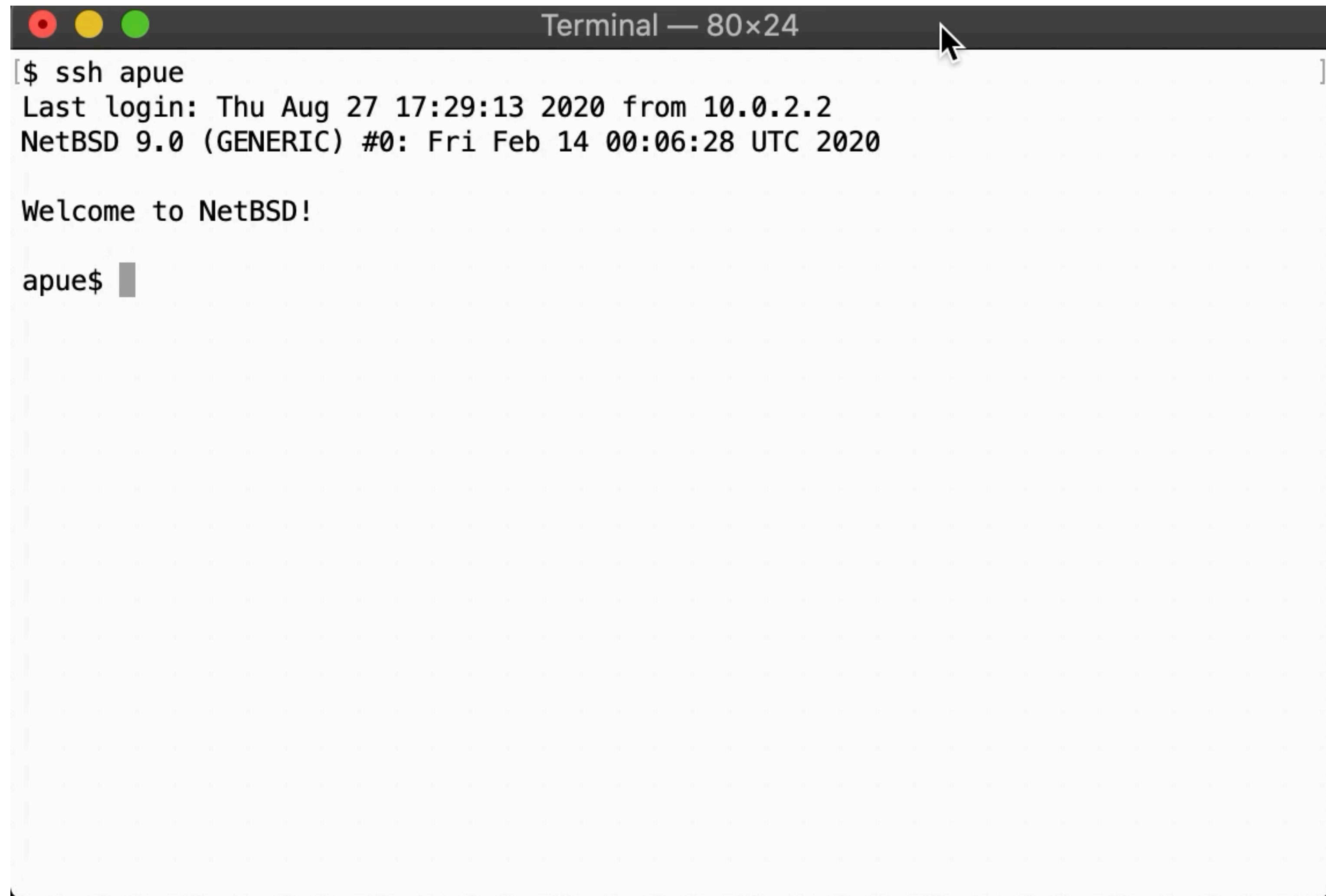
Welcome to NetBSD!

apue$
```



# Processes

- Programs executing in memory are called processes.
- Programs are brought into memory via one of the `exec(3)` / `execve(2)` functions.
- Each process is identified by a guaranteed unique non-negative integer called the *processes ID*.
- New processes can only be created via the `fork(2)` system call.
- Process control is performed mainly via the `fork(2)`, `exec(3)`, and `waitpid(2)` functions.

A terminal window titled "Terminal — 80x24" with a mouse cursor pointing at it. The terminal shows an SSH session starting with the command "\$ ssh apue". It displays the login history: "Last login: Thu Aug 27 17:29:13 2020 from 10.0.2.2" and "NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020". Below this is the message "Welcome to NetBSD!" and the prompt "apue\$ " with a cursor.

```
Terminal — 80x24
[$ ssh apue
Last login: Thu Aug 27 17:29:13 2020 from 10.0.2.2
NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020

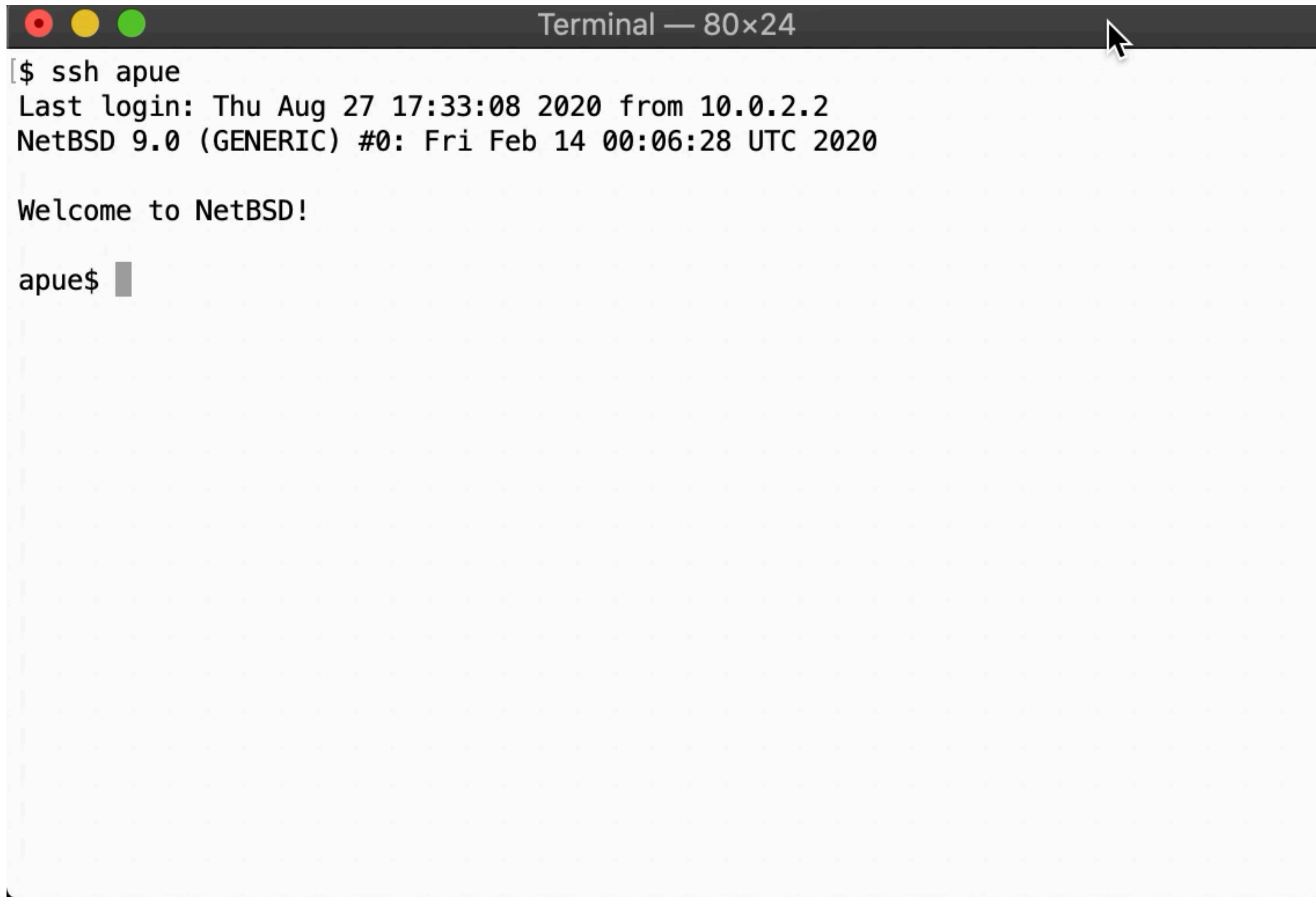
Welcome to NetBSD!

apue$
```

# Signals

Signals notify a process that a condition has occurred. Signals may be:

- allowed to cause the default action
- intentionally and explicitly ignored
- caught and control transferred to a user-defined function

A terminal window titled "Terminal — 80x24" with a dark gray title bar and three colored window control buttons (red, yellow, green) on the left. The terminal content shows an SSH session starting with the command "\$ ssh apue". It displays the login banner for NetBSD 9.0 (GENERIC) #0, including the last login time (Thu Aug 27 17:33:08 2020) and the date (Fri Feb 14 00:06:28 UTC 2020). The message "Welcome to NetBSD!" is shown, followed by the prompt "apue\$ " with a gray cursor block.

```
Terminal — 80x24
[$ ssh apue
Last login: Thu Aug 27 17:33:08 2020 from 10.0.2.2
NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020

Welcome to NetBSD!

apue$
```

## Homework

---

- setup your git repository for your class notes
- make sure your NetBSD VM is set up
- read `intro(2)` and `intro(7)`
- read chapters 1 & 2 in Stevens as well as the linked chapter on UNIX history and basics as you review Week 1
- review and repeat all the code exercises from this lecture segment
- read chapter 3 in Stevens to prepare for Week 2
- join the class Slack and say hi

Next week: File I/O and File Sharing