

# Threads in Applications

The background of the slide features a photograph of a classical statue, likely the 'The Spirit of the Law' by Jean-Louis Ponce, located in the Lincoln Park Conservatory in Chicago. The statue is a woman in a long, flowing dress, standing on a pedestal. The scene is set outdoors with many trees, some of which have yellow and orange leaves, suggesting an autumn setting. The entire image is overlaid with a semi-transparent red filter.

**CS 423 - University of Illinois**

Wade Fagen-Ulmschneider

(Slides built from Adam Bates and Tianyin Xu previous work on CS 423.)

# Threads

- ★ Threads are the lightest-weight method for separate execution paths on a CPU.

# Threads

- ★ Threads are the lightest-weight method for separate execution paths on a CPU.

## Advantages:

- ★ **Shared memory** between threads of the same process.
- ★ Kernel threads are **scheduled independently** by the OS.
- ★ **Lightest-weight** method separate execution. (Don't need to swap page table, etc.)

## Disadvantages

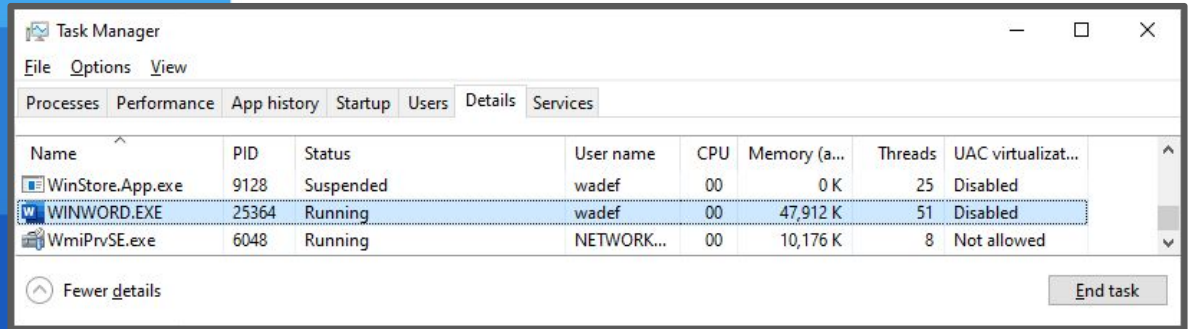
- ★ **Shared signals and exceptions:**
  - An error in a single thread results in the whole process being killed.
- ★ **Shared memory** between threads:
  - No isolation between threads (not suitable for running untrusted code).

# Threads are Particularly Useful For:

- ★ Making **blocking calls** within an interactive application,
- ★ Responses to **asynchronous events** (ex: “UI thread”, “net thread”),
- ★ **Long-running task** threads (ex: “worker thread”),
- ★ **Parallel algorithms/code** (ex: parallel sorts), and
- ★ Running tasks of **different priorities** within one process
  - Ex: `pthread_setschedparam`

# Example: Word Processor

Microsoft Word on Windows uses one process and many threads:



The screenshot shows the Windows Task Manager window with the 'Details' tab selected. The 'WINWORD.EXE' process is highlighted, showing it is running with PID 25364, using 47,912 K of memory, and has 51 threads. Other processes shown include WinStore.App.exe (Suspended) and WmiPrivSE.exe (Running).

Name	PID	Status	User name	CPU	Memory (a...)	Threads	UAC virtualizat...
WinStore.App.exe	9128	Suspended	wadef	00	0 K	25	Disabled
WINWORD.EXE	25364	Running	wadef	00	47,912 K	51	Disabled
WmiPrivSE.exe	6048	Running	NETWORK...	00	10,176 K	8	Not allowed

- ★ Only one process (pid=25364)
  - ...but **51 different threads!**



# Common Multithreaded Design Patterns

## ★ **Manager/Worker Pattern**

- One thread receives all requests and sends work to be completed by “worker threads”.
- *Example: Apache Web Server*

## ★ **Pipeline Pattern**

- Each thread handles a specific sub-operation in a series of processing steps
- *Example: Real-time video encoding*

## ★ **Other Patterns:** Peer Processing, ...and many more...



ALMA MATER

TO THE UNIVERSITY OF CALIFORNIA  
AT BERKELEY



# POSIX Interfaces



**CS 423 - University of Illinois**

Wade Fagen-Ulmschneider

(Slides built from Adam Bates and Tianyin Xu previous work on CS 423.)

# POSIX

- ★ The **Portable Operating System Interface (POSIX)** is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems.

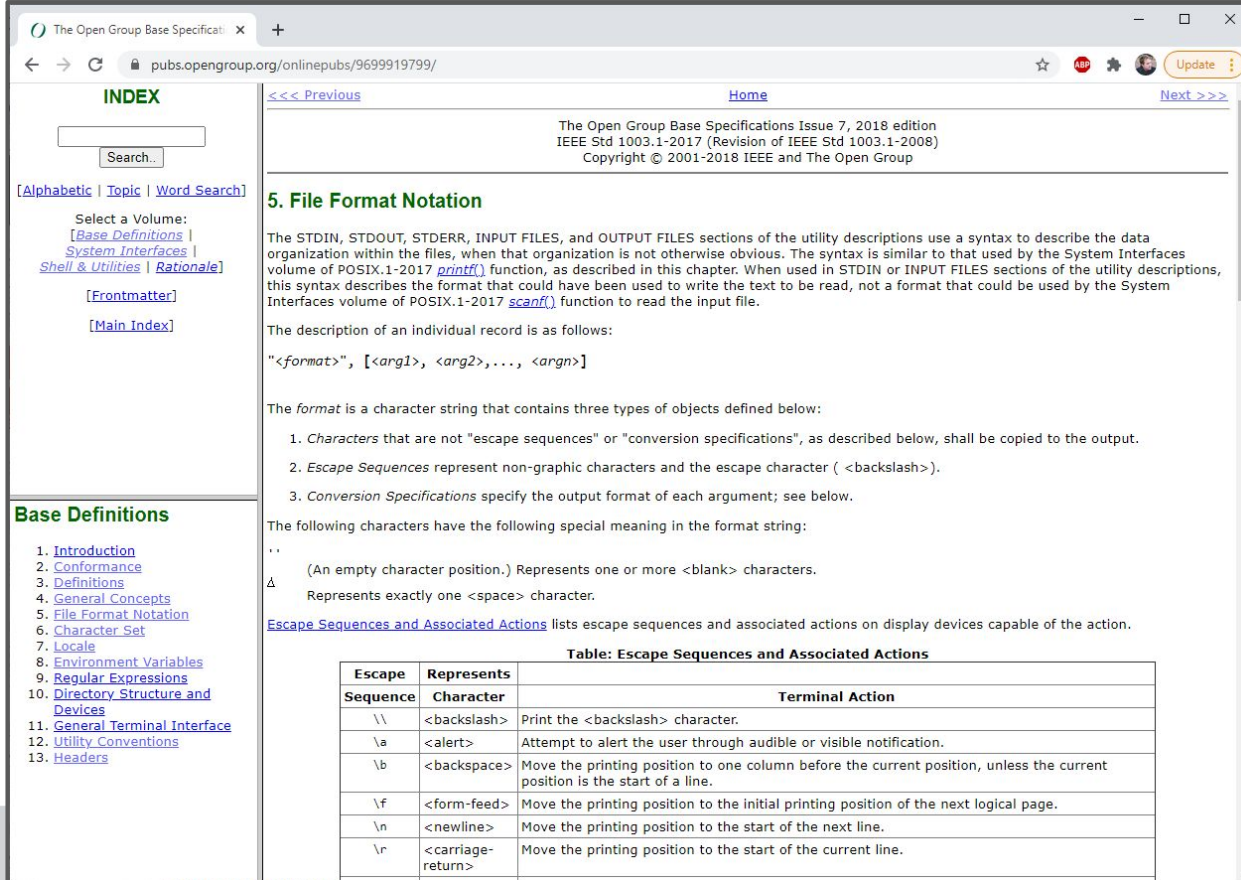
# POSIX

- ★ The **Portable Operating System Interface (POSIX)** is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems.
  - POSIX defines **SO MUCH** of what we consider Linux.

# POSIX

- ★ The **Portable Operating System Interface (POSIX)** is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems.
  - Four Volumes in the current standard (POSIX.1-2017):
    - Vol 1: “Base Definitions”
    - Vol 2: “System Interfaces”
    - Vol 3: “Shell & Utilities”
    - Vol 4: “Rationale”

# POSIX: printf format



The screenshot shows a web browser window displaying the POSIX.1-2007 online publication page for File Format Notation. The browser address bar shows the URL <https://pubs.opengroup.org/onlinepubs/9699919799/>. The page title is "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".

The page content is as follows:

## 5. File Format Notation

The STDIN, STDOUT, STDERR, INPUT FILES, and OUTPUT FILES sections of the utility descriptions use a syntax to describe the data organization within the files, when that organization is not otherwise obvious. The syntax is similar to that used by the System Interfaces volume of POSIX.1-2017 [printf\(\)](#) function, as described in this chapter. When used in STDIN or INPUT FILES sections of the utility descriptions, this syntax describes the format that could have been used to write the text to be read, not a format that could be used by the System Interfaces volume of POSIX.1-2017 [scanf\(\)](#) function to read the input file.

The description of an individual record is as follows:

```
"<format>", [<arg1>, <arg2>, ..., <argn>]
```

The *format* is a character string that contains three types of objects defined below:

1. *Characters* that are not "escape sequences" or "conversion specifications", as described below, shall be copied to the output.
2. *Escape Sequences* represent non-graphic characters and the escape character ( <backslash>).
3. *Conversion Specifications* specify the output format of each argument; see below.

The following characters have the following special meaning in the format string:

```
'' (An empty character position.) Represents one or more <blank> characters.  
Δ Represents exactly one <space> character.
```

[Escape Sequences and Associated Actions](#) lists escape sequences and associated actions on display devices capable of the action.

Escape Sequence	Represents Character	Terminal Action
\\	<backslash>	Print the <backslash> character.
\a	<alert>	Attempt to alert the user through audible or visible notification.
\b	<backspace>	Move the printing position to one column before the current position, unless the current position is the start of a line.
\f	<form-feed>	Move the printing position to the initial printing position of the next logical page.
\n	<newline>	Move the printing position to the start of the next line.
\r	<carriage-return>	Move the printing position to the start of the current line.



# POSIX: environmental variables

The screenshot shows a web browser window displaying the POSIX.1-2007 online publication page for environmental variables. The browser address bar shows the URL <https://pubs.opengroup.org/onlinepubs/9699919799/>. The page title is "The Open Group Base Specifications".

The page content is organized into several sections:

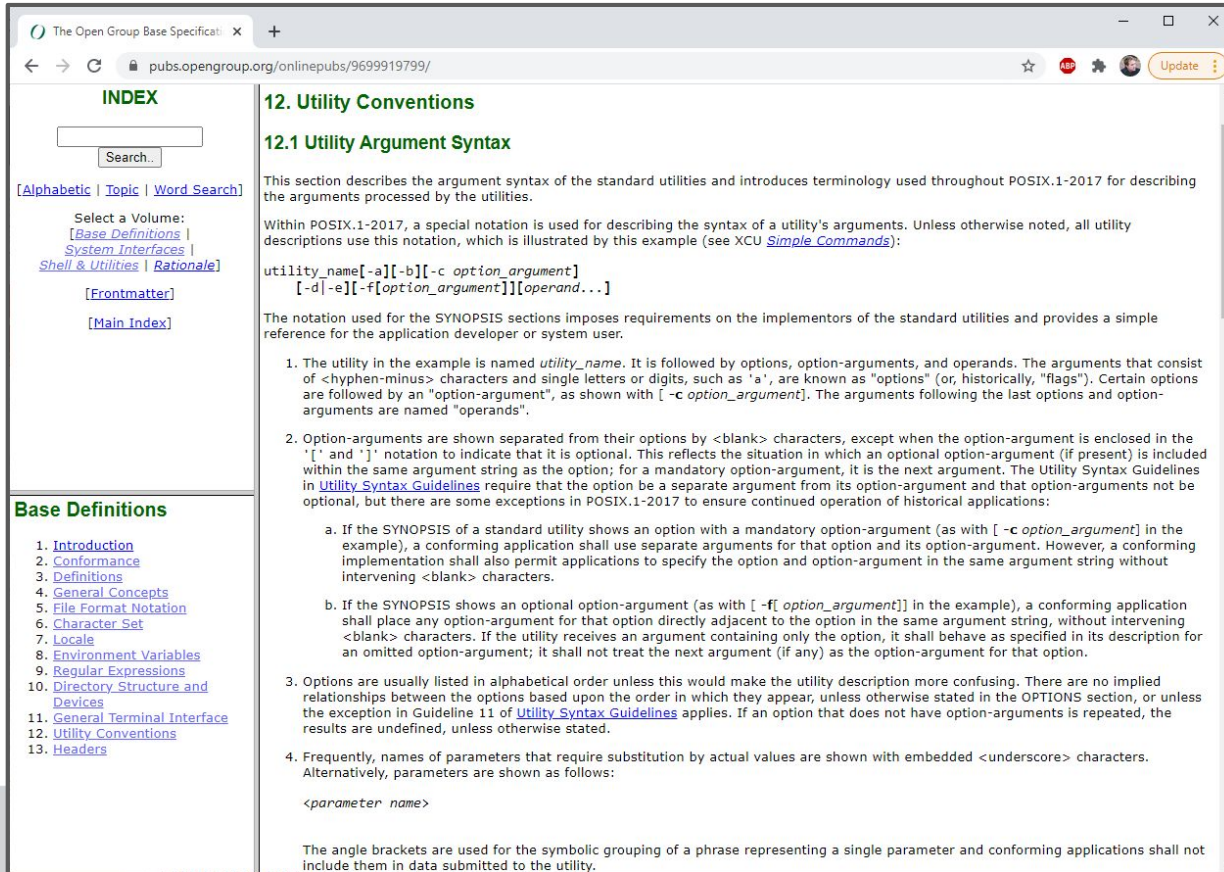
- INDEX**: A search box and navigation links for [Alphabetic], [Topic], [Word Search], [Base Definitions], [System Interfaces], [Shell & Utilities], [Rationale], [Frontmatter], and [Main Index].
- Base Definitions**: A list of 13 numbered links: 1. Introduction, 2. Conformance, 3. Definitions, 4. General 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.
- 8.1 Environment Variable Definition**:
  - Environment variables defined in this chapter affect the operation of multiple utilities, functions, and applications. There are other environment variables that are of interest only to specific utilities. Environment variables that apply to a single utility only are defined as part of the utility description. See the ENVIRONMENT VARIABLES section of the utility descriptions in the Shell and Utilities volume of POSIX.1-2017 for information on environment variable usage.
  - The value of an environment variable is a string of characters. For a C-language program, an array of strings called the environment shall be made available when a process begins. The array is pointed to by the external variable *environ*, which is defined as:

```
extern char **environ;
```
  - These strings have the form *name=value*; *names* shall not contain the character '='. For values to be portable across systems conforming to POSIX.1-2017, the value shall be composed of characters from the portable character set (except NUL and as indicated below). There is no meaning associated with the order of strings in the environment. If more than one string in an environment of a process has the same *name*, the consequences are undefined.
  - Environment variable names used by the utilities in the Shell and Utilities volume of POSIX.1-2017 consist solely of uppercase letters, digits, and the <underscore> ('\_') from the characters defined in [Portable Character Set](#) and do not begin with a digit. Other characters may be permitted by an implementation; applications shall tolerate the presence of such names. Uppercase and lowercase letters shall retain their unique identities and shall not be folded together. The name space of environment variable names containing lowercase letters is reserved for applications. Applications can define any environment variables with names from this name space without modifying the behavior of the standard utilities.
  - Note:** Other applications may have difficulty dealing with environment variable names that start with a digit. For this reason, use of such names is not recommended anywhere.
  - The *values* that the environment variables may be assigned are not restricted except that they are considered to end with a null byte and the total space used to store the environment and the arguments to the process is limited to {ARG\_MAX} bytes.
  - Other *name= value* pairs may be placed in the environment by, for example, calling any of the [setenv\(\)](#), [unsetenv\(\)](#), [\[XSI\]](#) [putenv\(\)](#) functions, assigning a new value to the *environ* variable, or by using *envp* arguments when creating a process; see *exec* in the System Interfaces volume of POSIX.1-2017.
  - If the application modifies the pointers to which *environ* points, the behavior of all interfaces described in the System Interfaces volume of POSIX.1-2017 is undefined.
  - It is unwise to conflict with certain variables that are frequently exported by widely used command interpreters and applications:

ARFLAGS	IFS	MAILPATH	PS1
CC	LANG	MAILRC	PS2
CDPATH	LC_ALL	MAKEFLAGS	PS3
CFLAGS	LC_COLLATE	MAKESHELL	PS4
CHARSET	LC_CTYPE	MANPATH	PWD
COLUMNS	LC_MESSAGES	MBOX	RANDOM
DATETIME	LC_MONETARY	MORE	SECONDS



# POSIX: Utility Program Conventions



The screenshot shows a web browser window displaying the POSIX.1-2017 online publication page for "12. Utility Conventions". The browser's address bar shows the URL <https://pubs.opengroup.org/onlinepubs/9699919799/>. The page has a left sidebar with an "INDEX" section containing a search box and navigation links like "[Alphabetic]", "[Topic]", and "[Word Search]". Below the index is a "Base Definitions" section with a numbered list of 13 items, including "Introduction", "Conformance", "Definitions", "General Concepts", "File Format Notation", "Character Set", "Locale", "Environment Variables", "Regular Expressions", "Directory Structure and Devices", "General Terminal Interface", "Utility Conventions", and "Headers".

## 12. Utility Conventions

### 12.1 Utility Argument Syntax

This section describes the argument syntax of the standard utilities and introduces terminology used throughout POSIX.1-2017 for describing the arguments processed by the utilities.

Within POSIX.1-2017, a special notation is used for describing the syntax of a utility's arguments. Unless otherwise noted, all utility descriptions use this notation, which is illustrated by this example (see XCU [Simple Commands](#)):

```
utility_name[-a][-b][-c option_argument]
[-d][-e][-f[option_argument]][operand...]
```

The notation used for the SYNOPSIS sections imposes requirements on the implementors of the standard utilities and provides a simple reference for the application developer or system user.

1. The utility in the example is named *utility\_name*. It is followed by options, option-arguments, and operands. The arguments that consist of <hyphen-minus> characters and single letters or digits, such as 'a', are known as "options" (or, historically, "flags"). Certain options are followed by an "option-argument", as shown with [-c *option\_argument*]. The arguments following the last options and option-arguments are named "operands".
2. Option-arguments are shown separated from their options by <blank> characters, except when the option-argument is enclosed in the '[' and ']' notation to indicate that it is optional. This reflects the situation in which an optional option-argument (if present) is included within the same argument string as the option; for a mandatory option-argument, it is the next argument. The Utility Syntax Guidelines in [Utility Syntax Guidelines](#) require that the option be a separate argument from its option-argument and that option-arguments not be optional, but there are some exceptions in POSIX.1-2017 to ensure continued operation of historical applications:
  - a. If the SYNOPSIS of a standard utility shows an option with a mandatory option-argument (as with [-c *option\_argument*] in the example), a conforming application shall use separate arguments for that option and its option-argument. However, a conforming implementation shall also permit applications to specify the option and option-argument in the same argument string without intervening <blank> characters.
  - b. If the SYNOPSIS shows an optional option-argument (as with [-f[ *option\_argument*]] in the example), a conforming application shall place any option-argument for that option directly adjacent to the option in the same argument string, without intervening <blank> characters. If the utility receives an argument containing only the option, it shall behave as specified in its description for an omitted option-argument; it shall not treat the next argument (if any) as the option-argument for that option.
3. Options are usually listed in alphabetical order unless this would make the utility description more confusing. There are no implied relationships between the options based upon the order in which they appear, unless otherwise stated in the OPTIONS section, or unless the exception in Guideline 11 of [Utility Syntax Guidelines](#) applies. If an option that does not have option-arguments is repeated, the results are undefined, unless otherwise stated.
4. Frequently, names of parameters that require substitution by actual values are shown with embedded <underscores> characters. Alternatively, parameters are shown as follows:

```
<parameter name>
```

The angle brackets are used for the symbolic grouping of a phrase representing a single parameter and conforming applications shall not include them in data submitted to the utility.

# POSIX: Utility Program Conventions

- ★ Utility names should be between two and nine characters, inclusive.
- ★ Utility names should include lowercase letters (the **lower** character classification) and digits only from the portable character set.
- ★ Each option name should be a single alphanumeric character (the **alnum** character classification) from the portable character set. The -W (capital-W) option shall be reserved for vendor options. Multi-digit options should not be allowed.



# POSIX: Utility Program Conventions

- ★ All options should be preceded by the '-' delimiter character.
- ★ One or more options without option-arguments, followed by at most one option that takes an option-argument, should be accepted when grouped behind one '-' delimiter.
- ★ ...and 9 others...

# POSIX: System Interfaces

★ POSIX.1-2017 Volume 2 defines **1,191** different system interfaces!

*...here's just 60 of them:*

<code>a64l()</code>	<code>fmodl()</code>	<code>pthread_cond_broadcast()</code>	<code>wcwidth()</code>
<code>abort()</code>	<code>fmtmsg()</code>	<code>pthread_cond_destroy()</code>	<code>wmemchr()</code>
<code>abs()</code>	<code>fnmatch()</code>	<code>pthread_cond_init()</code>	<code>wmemcmp()</code>
<code>accept()</code>	<code>fopen()</code>	<code>pthread_cond_signal()</code>	<code>wmemcpy()</code>
<code>access()</code>	<code>fork()</code>	<code>pthread_cond_timedwait()</code>	<code>wmemmove()</code>
<code>acosf()</code>	<code>fpathconf()</code>	<code>pthread_cond_wait()</code>	<code>wmemset()</code>
<code>acoshf()</code>	<code>fpclassify()</code>	<code>pthread_create()</code>	<code>wordexp()</code>
<code>acoshl()</code>	<code>fprintf()</code>	<code>pthread_detach()</code>	<code>wordfree()</code>
<code>acos()</code>	<code>fputc()</code>	<code>pthread_equal()</code>	<code>wprintf()</code>
<code>acosl()</code>	<code>fputs()</code>	<code>pthread_exit()</code>	<code>write()</code>
<code>aio_cancel()</code>	<code>fputwc()</code>	<code>pthread_getconcurrency()</code>	<code>writev()</code>
<code>aio_error()</code>	<code>fputws()</code>	<code>pthread_getcpuclockid()</code>	<code>wscanf()</code>
<code>aio_fsync()</code>	<code>fread()</code>	<code>pthread_getschedparam()</code>	<code>y0()</code>
<code>aio_read()</code>	<code>freeaddrinfo()</code>	<code>pthread_getspecific()</code>	<code>y1()</code>
	<code>free()</code>	<code>pthread_join()</code>	<code>yn()</code>

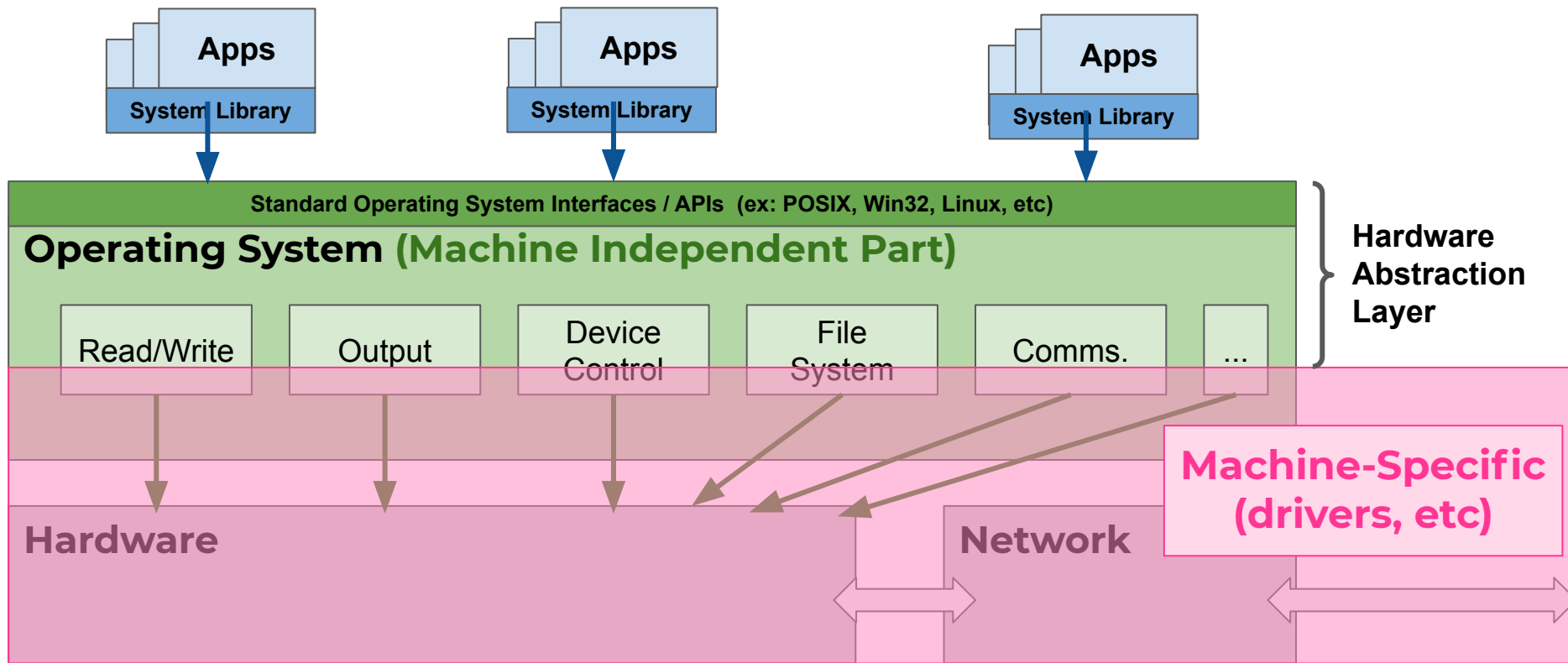


# POSIX: Utility Programs

★ POSIX.1-2017 Volume 3 defines **160** different utility programs:

*...here the first 120 of the 160 utility programs:*

admin	chgrp	dd	file	ipcs	make	pathchk	qsig
alias	chmod	delta	find	jobs	man	pax	qstat
ar	chown	df	fold	join	mesg	pr	qsub
asa	cksum	diff	fort77	kill	mkdir	printf	read
at	cmp	dirname	fuser	lex	mkfifo	prs	renice
awk	comm	du	gencat	link	more	ps	rm
basename	command	echo	get	ln	mv	pwd	rmdel
batch	compress	ed	getconf	locale	newgrp	qalter	rmdir
bc	cp	env	getopts	localedef	nice	qdel	sact
bg	crontab	ex	grep	logger	nl	qhold	sccs
c99	csplit	expand	hash	logname	nm	qmove	sed
cal	ctags	expr	head	lp	nohup	qmsg	sh
cat	cut	false	iconv	ls	od	qrerun	sleep
cd	cxref	fc	id	m4	paste	qrls	sort
cflow	date	fg	ipcrm	mailx	patch	qselect	split





ALMA MATER

TO THE UNIVERSITY OF CALIFORNIA  
AT BERKELEY

# Hourglass Model for a Systems Programming Interfaces

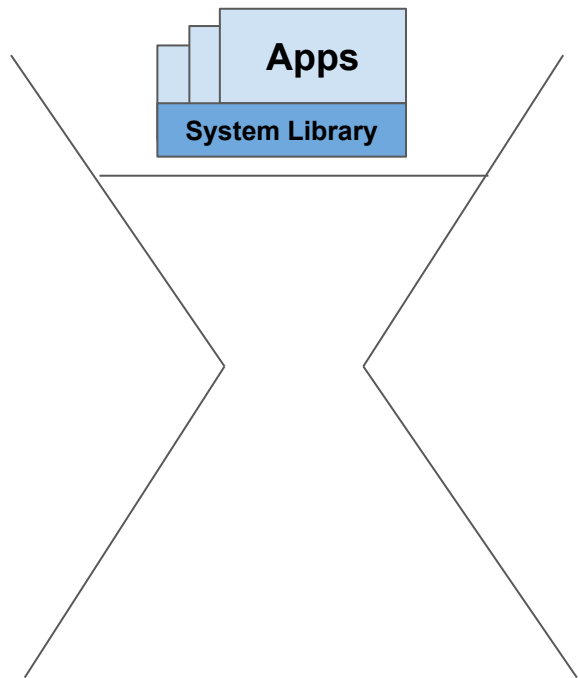
The background of the slide features a classical statue, likely representing a personification of Justice or Liberty, with arms outstretched. The entire image is overlaid with a semi-transparent red filter. The statue is positioned in the lower right quadrant, partially obscured by the text.

**CS 423 - University of Illinois**

Wade Fagen-Ulmschneider

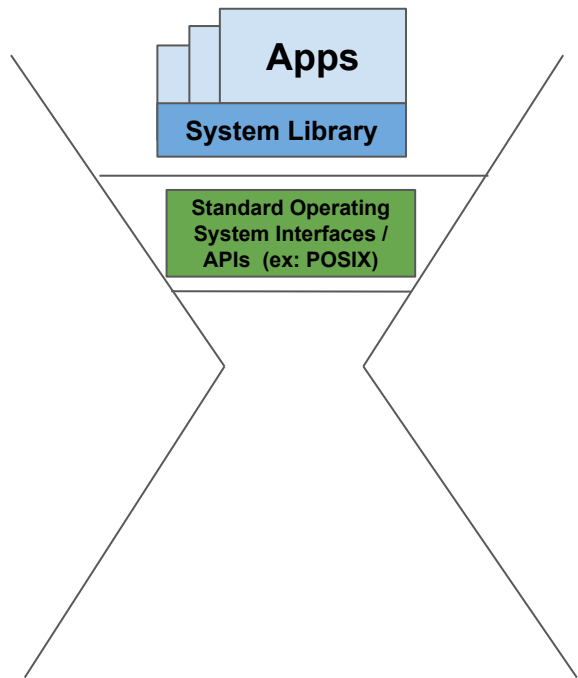
(Slides built from Adam Bates and Tianyin Xu previous work on CS 423.)

# POSIX: “Hourglass Model”



- ★ Millions of different apps, all doing different things:
  - Web Browser
  - Compiler
  - Word Processing
  - Database
  - Web Server
  - Minecraft
  - ...etc...

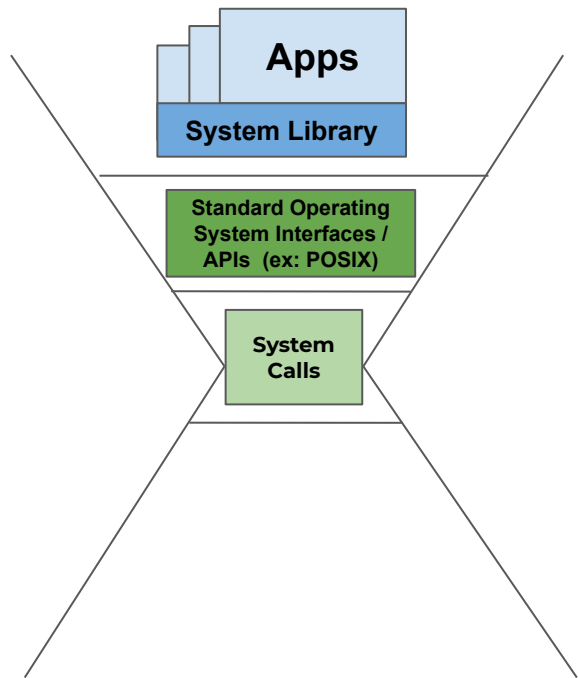
# POSIX: “Hourglass Model”



- ★ Hundreds to thousands of APIs to interact with the OS:
  - System Library API interact with the POSIX interface APIs.
  - *Remember, POSIX.1-2017 has 1,191 different APIs!*

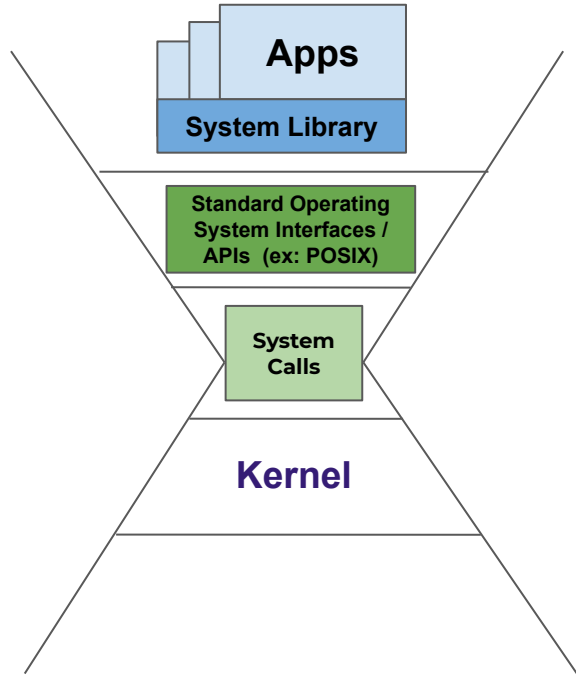


# POSIX: “Hourglass Model”



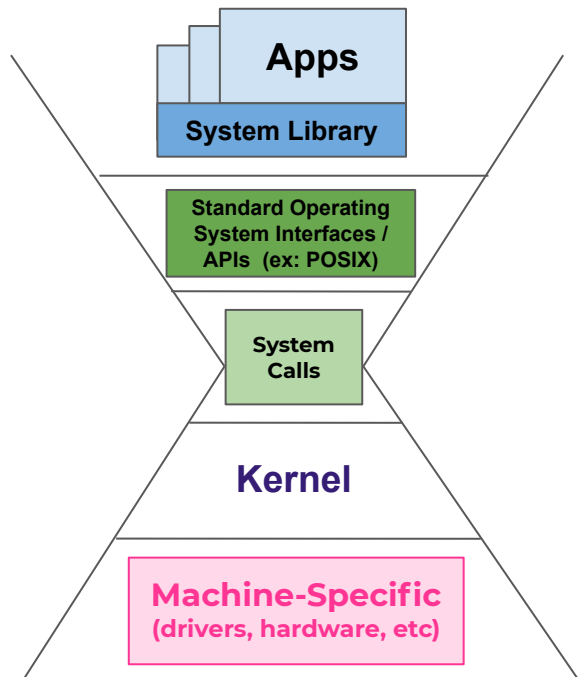
- ★ Smallest interface is the syscalls:
  - Only 461 system calls documented for Linux Kernel 5.10.  
<https://man7.org/linux/man-pages/man2/syscalls.2.html>
  - *Over 100 of them are removed or hardware platform specific.*

# POSIX: “Hourglass Model”



- ★ Kernel sits between the system calls and the hardware.
- ★ We'll be programming code here all semester!

# POSIX: “Hourglass Model”



- ★ Just like apps, there is **a lot** of hardware out there!
  - New processors, network interfaces, graphics cards, etc every year.

# Python Code:

App (Python): `open(...)`



Python is written in C ("CPython"), making a call to the POSIX library call...

C: `fopen(...)`

When compiled on  
Win32 system...

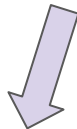


SysCall: `CreateFileA(...)`

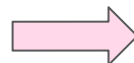
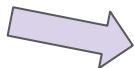


When compiled on a  
Linux system....

SysCall: `open(...)`



Kernel



Machine-Specific  
(drivers, hardware, etc)



ALMA MATER

TO THE UNIVERSITY OF CALIFORNIA  
BERKELEY