

Applications

Complex applications must:

- run for weeks or months
- properly release resources to avoid waste
- cope with outrageously malicious user input
- recover from errors and continue running

POSIX

Portable Operating System Interface (POSIX)

- an important step toward producing reliable applications
- POSIX compliant systems no longer need to contend with small but critical variations in behavior of library functions across platforms

Objectives

- Learn how OS manages resources
- Experiment with buffer overflows
- Explore concurrency and asynchronous behavior
- Strengthen basic OS terminology
- Understand serious implications of incorrect code

Asynchronous Operation

- Computer system events occur at unpredictable times and in unpredictable order
- Programs must work for all possible orderings

Concurrency

- Sharing of resources in the same time frame
- Apparent concurrency is sharing the same CPU, memory, or I/O device
- Real concurrency is sharing the same program among several CPUs, memories, and/or I/O devices

Communication

- The conveying of information from one entity to another
- Network communication introduces a myriad of new problems resulting in unpredictable times and possible remote failures

Safe Functions

- Thread-Safe – Can be invoked concurrently or by multiple threads.
- Async-Signal-Safe – Can be called without restriction from a signal handler.

These terms replace the older notion of reentrant function.

CPU Events Relative to Real Time

Item	Time	Scaled Time in Human Terms (2 billion times slower)
Processor Cycle	0.5 ns (2GHZ)	1 second
Cache Access	1 ns	2 seconds
Memory Access	15ns	30 seconds
Context Switch	5,000ns (5 μ s)	167 minutes
Disk Access	7,000,000ns (7 ms)	162 days
Time Quantum	100,000,000ns (100ms)	116 days

Screen Filling Comparisons

Modem type	Bits per second	Time needed to display	
		Text	Graphics
1979 telephone modem	300	1 min	6 hours
1983 telephone modem	2,400	6 secs	45 mins
current telephone modem	57,600	0.28 secs	109 secs
current DSL modem	768,000	0.02 secs	8 secs

Interrupts

- Causes transfer of control to interrupt handling routine
- Synchronous interrupts are invoked by program system calls
- Asynchronous interrupts are invoked by external devices such as I/O or timer

Signals

- Notifies software of an event
- Signals are often invoked by interrupt handling routine
- A signal is caught if the process receiving the signal executes an interrupt handling routine (signal handler) for the signal

Processes

- Concurrent processes are invoked by fork
- Processes with common ancestor can communicate through pipes
- Processes without a common ancestor can communicate by signals, semaphores, shared address space, or messages

Threads

- Multiple threads of execution can provide concurrency within a process
- The stream of instructions is called the program's thread of execution
- If two distinct threads of execution share a resource within a time from, care must be taken that these threads do not interfere with each other
- A thread standard has been incorporated in POSIX

Buffer Overflow

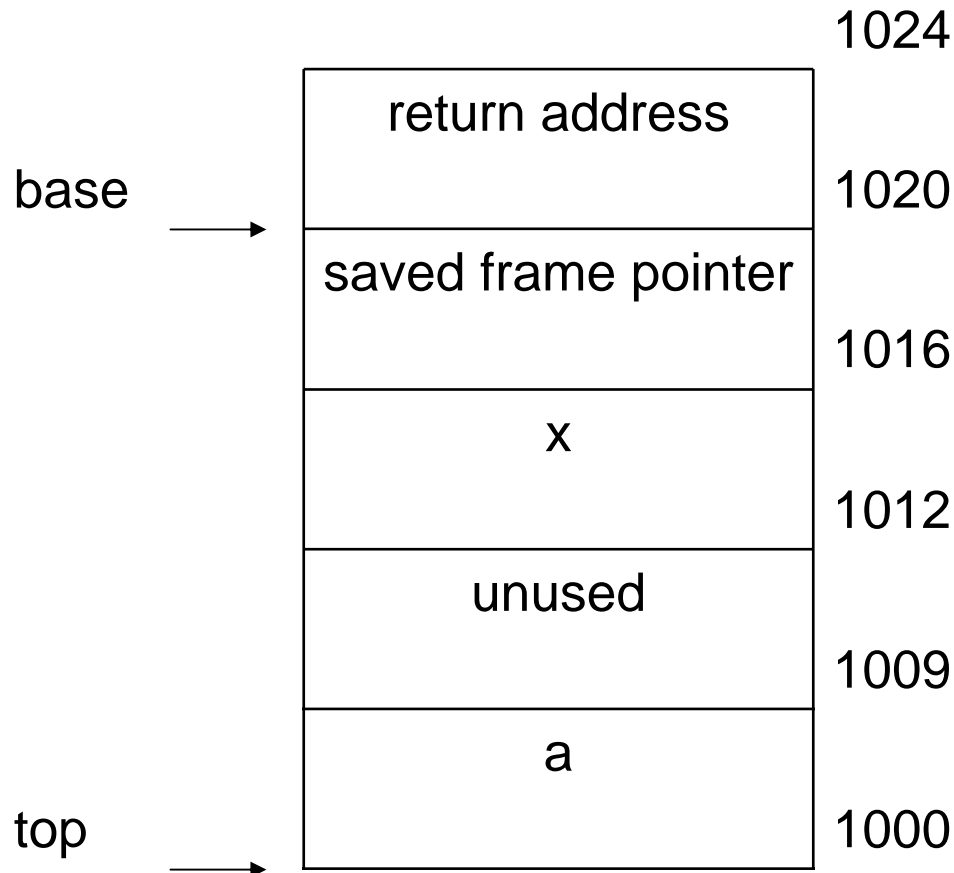
```
char buf[80];  
printf("Enter your first name");  
scanf("%s", buf);
```

```
char buf[80];  
printf("Enter your first name");  
scanf("%79s", buf);
```

Password Program

```
#include <stdio.h>
#include <string.h>
int checkpass(void) {
    int x;
    char a[9];
    x=0;
    fprintf(stderr,"a at %p and\nx at %p\n", (void) a, (void *)&x);
    printf("Enter a short word: ");
    scanf("%s", a);
    if (strcmp (a, "mypass") == 0) x=1;
    return x; }
```

Stack Representation



Stack Conditions

- 12 bytes are allocated for array a even though only 9 are needed so a is aligned to word memory.
- Integers and Pointers are 4 bytes.

Stack Problems

- If the user enters 12 characters, the string overwrites 1 byte of x without changing its value
- If the user enters more than 12 characters, x is overwritten changing its value
- If the user enters a long password, the return address is overwritten – the function may try to return to address space outside the program causing a segmentation fault

Buffer Overflow and Worms

- Morris worm exploited buffer overflow in the finger daemon
- Forced many system administrators to disconnect sites from the Internet

Telnet and Buffer Overflow

- Buffer overflow occurs if password is too long
- Hackers purposely type in long password to overwrite memory
- The idea is to overwrite the return value so they get in even though the password is incorrect

UNIX Standards

- ANSI C
- POSIX
- Spec 1170
- ISO C

POSIX Extensions

code	extension	Solaris 9
AIO	asynchronous input and output	yes
CX	ISO C standard extension	yes
FSC	file synchronization	yes
RTS	real time signals	yes
SEM	semaphores	yes
THR	threads	yes
TMR	timers	yes
TPS	thread execution scheduling	yes
TSA	thread stack address attribute	no
TSF	thread-safe functions	yes
XSI	XSI extension	yes

POSIX

- POSIX-compliant implementation must support the POSIX standard
- Table E.1 in appendix E lists all extensions to the base standard
- POSIX-compliant implementations have the symbol `_POSIX_VERSION` defined in the include file *unistd.h*