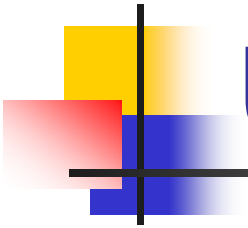
Computer Network Programming

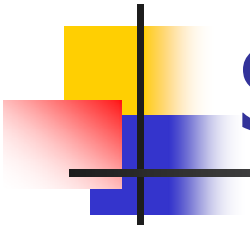
UNIX Signals

Dr. Sam Hsu
Computer Science & Engineering
Florida Atlantic University



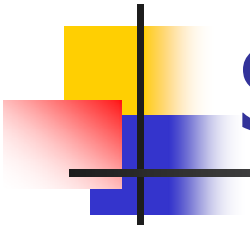
UNIX Signals

- Signal Generation
- Use of Signals
- Signal Status
- Signal Disposition
- Various Signal System Calls
- Signal Sets
- Reentrant Functions
- Slow System Calls
- Network Programming Tips



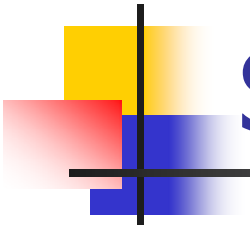
Signals

- A primitive way of doing IPC.
  - Are used to inform processes of asynchronous events.
  - An asynchronous event either terminates a process or is simply being ignored.
  - Arrangements can be made to trap signals.



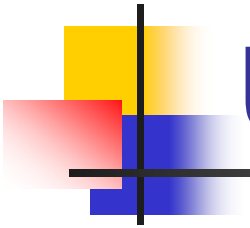
Signal Generation

- A signal is generated when (not a complete list):
  - A hardware exception occurs.
  - Interrupt or quit from control terminal.
  - An alarm timer expires.
  - A call to kill().
  - Termination of a child process.



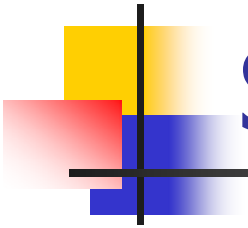
Signals Are Software Interrupts

- Each signal has a name.
  - A signal is identified by a named constant (symbolic constant).
  - A set of predefined numbers: 1~MAXSIG.
  - Details in signal.h.
- Signals may not be queued.
  - Implementation-dependent to recognize multiple instances of a signal.
- Order of service is not defined when different signals are pending on a process.



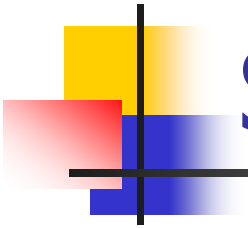
Use of Signals

- Intraprocess
- Interprocesses
  - With the same UID.
- Between kernel to any process.



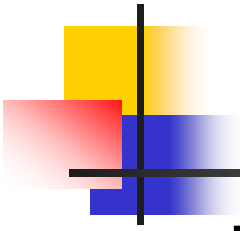
Signal Status

- A signal is said to be:
  - *generated* when the event that causes the signal occurs.
  - *delivered* when the action for a signal is taken.
  - *pending* during the time between the generation of the signal and its delivery.
  - *blocked* if unable to deliver due to a signal mask bit being set for the signal.



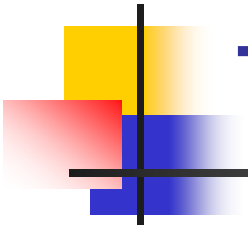
Sending Signals

- Signals can be sent to processes at any time.
  - By the kernel, or
  - A call to kill() by the user.
- However, signals are checked only when a process is about to return from the kernel mode to the user mode.



Signal Disposition

- Response to a signal, known as the *disposition* of the signal, can be one of the following:
  - Default action (SIG_DFL)
    - Termination in general.
  - Ignored (SIG_IGN)
    - Never posted to the process.
  - User-defined action
    - Needs a user-defined *signal handler*, or *signal-catching function*.
    - Most signals can be caught, or ignored except SIGKILL and SIGSTOP.

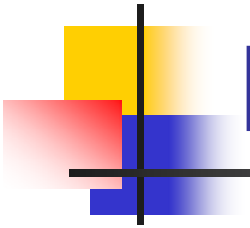


The signal() System Call

- Syntax

  void (*signal(int *signo*, void (**handler*)(int)))(int)

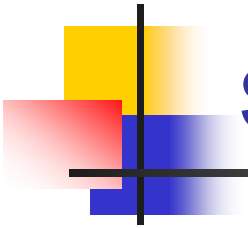
- Return value
  - The previous signal handler if OK, SIG_ERR on error.
- Early implementation of signal() was said to be unreliable.
  - Signals could get lost.
  - The signal handler for a signal was reset to default each time the same signal occured.
  - A process was unable to block signals.



Newer Versions of signal()

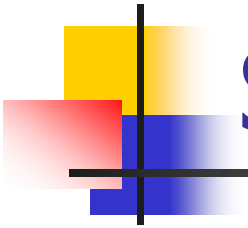
- signal() is replaced by sigset() in newer versions of UNIX SV for reliability.
  - Syntax

    void (*sigset(int *signo*, void (**handler*)(int)))(int)
- Has been further superseded by sigaction() in the latest implementations of various versions of UNIX systems.



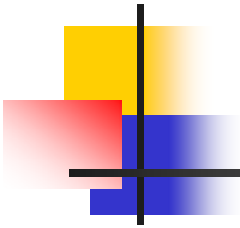
sigaction() Semantics

- A signal handler remains installed until uninstalled/changed.
- The delivery of a signal is blocked when its signal handler is being executed.
  - Additional signals may be blocked via sa_mask.
- If a signal gets generated one or more times while it is blocked, it may get delivered at most one time in general, after the signal is unblocked.
- One may impose process-wide signal blocking/unblocking using sigprocmask().



Signal Masks

- A signal mask is used to block signal delivery.
  - A blocked signal depends on the recipient process to unblock and handle it accordingly.
- A signal mask may be implemented using an integer.
  - Positional – each bit corresponds to one signal.
  - Bit ***1***'s – the corresponding signals are being blocked.
  - One problem – the number of different signals can exceed the number of bits in an integer.
- A process may query or change its signal mask by a call to sigprocmask().



Signal Sets

- Are used to represent multiple signals the number of which may exceed the number of bits in an integer.
- To manipulate signal sets, a new data type known as sigset_t with the following five predefined functions is specified in POSIX.1:

  sigemptyset()

  sigfillset()

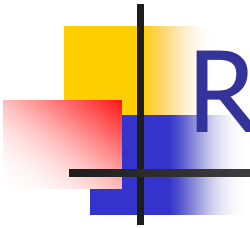
  sigaddset()

  sigdelset()

  sigismember()

Some Other Properties of Signals

- Signal dispositions are inherited by child.
- All signals are reset to default upon exec() unless ignored.
- Keyboard interrupts are ignored in background processes.

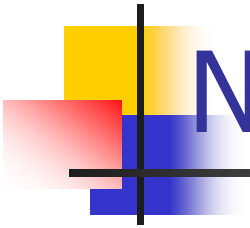


Reentrant Functions

- A function is considered to be *reentrant* if it can be reentered (called again) before a previous call finishes without causing any side effects.
  - No global data sharing for reentrant functions.
  - No *static* data structures.
- Try to avoid calling non-reentrant functions in a signal handler.

Slow System Calls

- A system call is considered *slow* if it can be blocked for an undetermined period of time. For example,
  - Terminal I/O
  - pause() and wait()
- A slow system call, in general, returns when a signal is caught and the signal handler returns.
  - This system call is said to be *interrupted*.
  - The interrupted system call returns *–1* with errno set to EINTR.



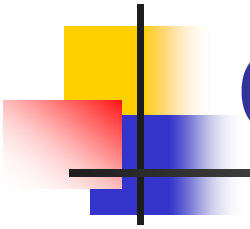
Network Programming Tips

- Need to catch SIGCHLD in parent before fork()ing.
- Need to avoid zombies by using waitpid() correctly in a SIGCHLD handler.
- Need to handle interrupted system calls when catching signals.

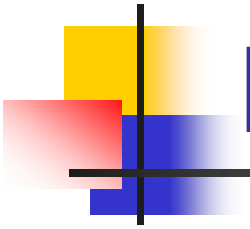
Some Relevant System Calls/Functions (1/2)

- kill()/raise()
  - kill() sends a signal to a process or a group of process.
  - raise() sends a signal to the calling process itself.
- alarm()
  - Is used to set a timer that will expire at a specified time in the future.
- pause()
  - Is used to suspend the calling process until a signal is received.

Some Relevant System Calls/Functions (2/2)



- sigpending()
  - Returns the set of signals that are blocked from delivery and currently pending for the calling process.
- sigsuspend()
- sigsetjmp()/siglongjmp()



Recommended Reading

- Read Chapter 10, *Advanced Programming in the UNIX Environment*, by W. Richard Stevens.