Signals

Stanley Yao
Computer Science Department
University of Arizona

Problems

- There is a very complicated computation that takes a very very long time, in order to let the user know the program is working hard on the computation instead of hanging, we want to print a "." on the screen every second while doing the computation.

Problems (cont.)

- In your program, you want to do some computation and also want to receive socket packages coming from the network. You don't want your program stupidly sitting there solely listening to the socket port. Instead, you want it to do the computation while there is no socket packages coming in, and whenever a package comes in you put the computation aside for a while and process the package first.

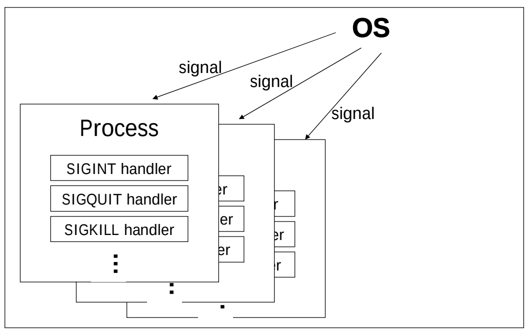
Signal

- A UNIX mechanism to notify a process about program faults and external events.
  - Program faults: divided by zero, segmentation violation, bus error, etc.
  - External events: process termination, I/O available, broken pipe, etc.
- Signals: Many different signals identified by a unique integer and represent a particular event
- Signal handler: in a process, for each signal, there is a function that is called automatically by the system when the signal is received by the process. You can register your own singal handler, otherwise a default one will be used.

Signal

Signal	Number	Default Action	Comment
SIGINT	2	Terminate	^C
SIGQUIT	3	Terminate & dump core	^\
SIGKILL	9	Terminate, can't handle or ignore	Kill -9
SIGSEGV	11	Terminate	Segmentation violation
SIGTERM	15	Terminate	Kill
SIGUSR1	Varies	Terminate	Program-defined meaning
SIGALRM	14	Terminate	Alarm signal
SIGCHLD	Varies	Ignored	Child stopped or terminate
SIGIO	Varies	Terminate	I/O now available

Big Picture



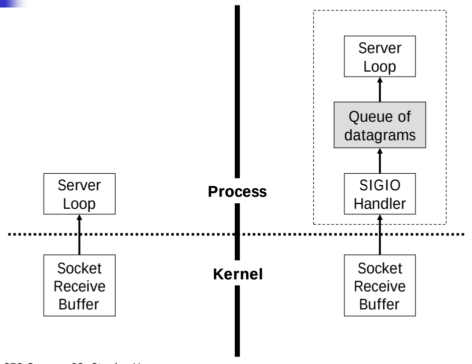
Setting Signal Handler

- typedef void (*sighandler_t)(int);
- sighandler_t signal(int signum, sighandler_t handler);
  - Install a new signal handler for the specified signal and returns the previous signal handler.
  - handler
    - SIG_IGN: ignore
    - SIG_DFL: default action

Signal Mask

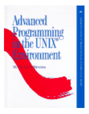
- int sigprocmask(int how, const sigset_t *set, sigset_t *oset);
  - Change the signal mask and return the old mask by oset (if oset is not NULL)
  - how
    - SIG_BLOCK: add set into the mask
    - SIG_UNBLOCK: remove set from the mask
    - SIG_SETMASK: replace the current mask with set

Signal-driven Socket

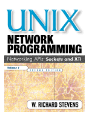


Further Readings

- Signals
  - Richard Stevens, *Advanced Programming in the UNIX Environment*, Addison-Wesley, 1992, ISBN 0-201-56317-7.
  - Sample programs: http://www.kohala.com/start/apue.html



- Sockets
  - Richard Stevens, *UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI*, Prentice Hall, 1998, ISBN 0-13-490012-X.
  - Sample programs: http://www.kohala.com/start/unpv12e.html



Acknowledgement

- John H. Hartman, *Classnotes for Csc352-Spring03*, CS Dept., University of Arizona, 2003
- Gail Anderson, Paul Anderson, *The Unix C Shell Field Guide*, Prentice Hall, 1986
- Richard Stevens, *UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI*, Prentice Hall, 1998, ISBN 0-13-490012-X