Signals

- Learning Objectives:
  - Explain what signals are
  - Use the system calls related to sending, blocking, unblocking signals.
  - Write a signal handler

What are Signals?

- A way for processes and the OS to interrupt other processes.
- A (very) small message notifying a process of some system event.
- The software equivalent to hardware interrupts.
- `man signal` will show you the `signal` library call as well as a list of the standard UNIX signals.
- Check out the whole list, but here are some of the "favorites."

ID	Name	Default Action	Event
2	SIGINT	Terminate	Interrupt (control-C)
9	SIGKILL	Terminate	Kill program
11	SIGSEGV	Terminate and dump core	Segmentation violation
14	SIGALRM	Terminate	Timer Signal
17	SIGCHLD	Ignore	Child stopped or terminated
19	SIGSTOP	Stops process	Process asked to stop

What exactly happens on a signal?

- Two distinct steps to transfer a signal:
  1. OS delivers signal to destination process, because
     - Some system event occurred, or
     - Signal delivery was explicitly requested via kill function
  2. Process receives signal (i.e., forced by OS to react to signal in some way)
     - Process can react in one of three ways:
       1. Ignore signal (i.e., do nothing)
       2. Terminate (maybe dumping core)
       3. Catch a signal with a signal handler function

Signal Status and Behavior

- Signal sent, but not received: pending
  - A process can only have one pending signal of each type pending at any point in time.
  - Signals are **not** queued – they are dropped!
  - Example:
    - You have two children.
    - Both exit, before you can receive either of the signals – you will receive only one SIGCHLD, not two!
- A process can block receipt of a signal
  - Blocked signals will be pending until the signal is unblocked.
  - Blocking signals is different from ignoring them.
- A signal can be received only once

Default Actions

- Each signal type has a predefined default action, one of:
  - The process terminates
  - The process terminates and dumps core
  - The process stops
    - until restarted by a SIGCONT signal
  - The process ignores the action
- Processes change default actions via:
  - `signal` system call
  - Signal handlers
  - Blocking and unblocking signals

Signal System Call & Signal Handlers

```
void signal(int signum, handler_t *handler)
```

- Overrides default action for signals of kind signum

- Handler can take on different values:
  - SIG_IGN: ignore signals of type signum
  - SIG_DFL: revert to the default action for signals of type signum
  - Otherwise, handler is a function pointer for a signal handler
- Signal handler is the function called upon receipt of a signal of type signum.
  - Referred to as installing handler
- Handler execution is called handling or catching signal
  - When handler function returns, control flow of interrupted process continues where it was when it was interrupted.

```
signals [31] cat signal-demo.c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int signal_received;

void
int_handler(int sig) {
    signal_received = sig;
}

int
main() {
    // Install signal handler.
    signal(SIGINT, int_handler);

    while (1) {
        sleep(2);
        printf("Process woke up, signal_received=%d\n", signal_received);
        signal_received = 0;
    }
}
signals [32]
```

Blocking Signals

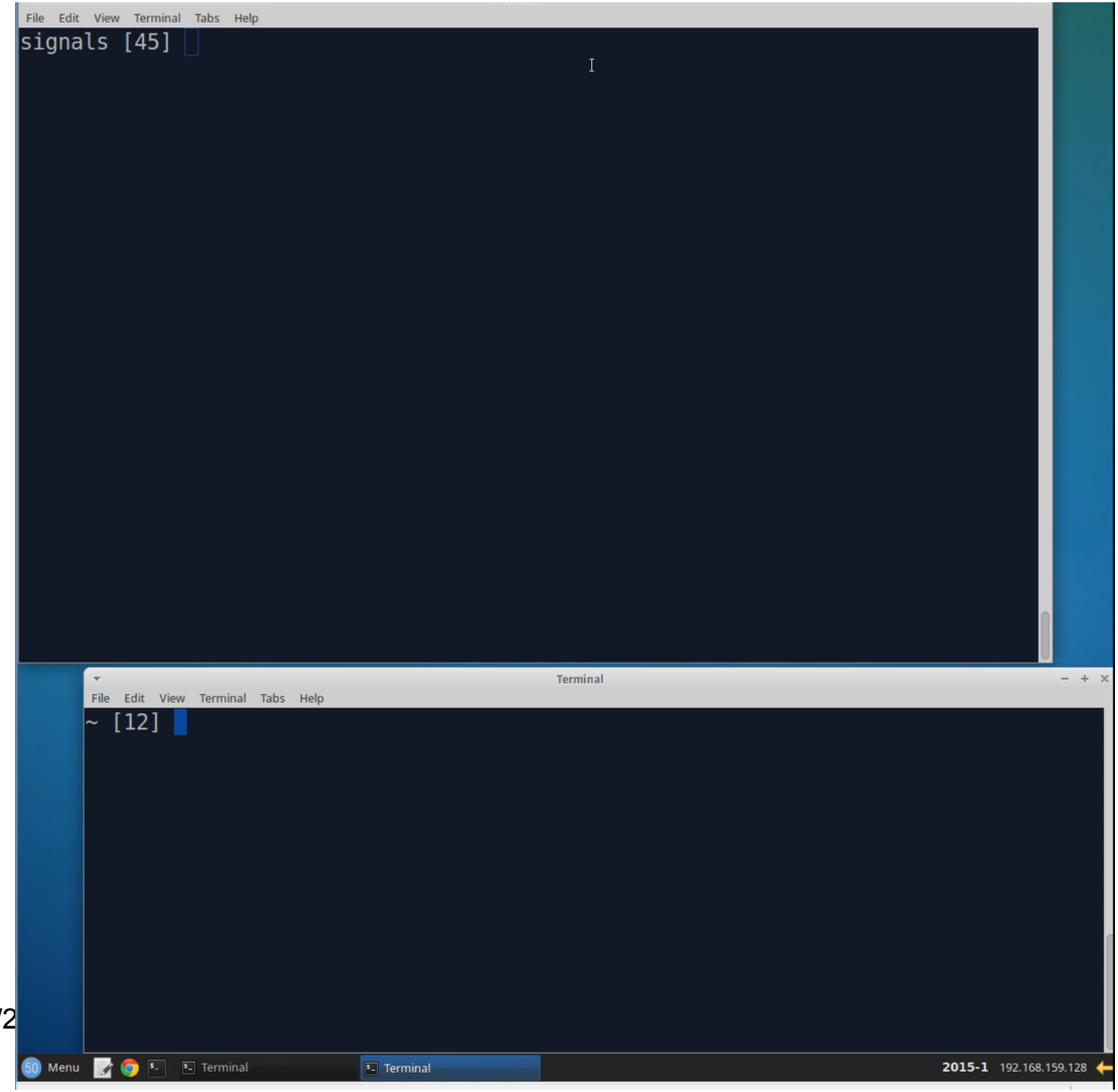
```
int sigprocmask(int how,
const sigset_t *restrict set, sigset_t *restrict oset);
```

- Allows a process to specify which signals it will receive (within reason).
  - Cannot block SIGKILL or SIGSTOP
  - The kernel maintains state indicating which signals a process is willing to receive. This call changes that state.
- `how` parameter:
  - Specifies what you're doing with this call:
    - SIG_BLOCK: add signals specified by set to the set of blocked signals.
    - SIG_UNBLOCK: remove signals specified by set from the set of blocked signals.
    - SIG_SETMASK: replace the set of blocked signals with those specified in set.

Blocking Signals (2)

```
int sigprocmask(int how,
const sigset_t *restrict set, sigset_t *restrict oset);
```

- `oset` parameter:
  - If not NULL, upon return, contains the value of the signal mask.
  - Lets you check the current state of the signals mask.
- `set` parameter:
  - A mask (one bit per signal) indicating for which signals, we want to modify behavior
- Helper functions
  - int sigemptyset(sigset_t *set) – initializes set to empty
  - int sigaddset(sigset_t *set, int signum) – adds the signal signum to the set
  - int sigdelset(sigset_t *set, int signum) – removes the signal signum from the sit.


File Edit View Terminal Tabs Help
signals [45]
Terminal
File Edit View Terminal Tabs Help
~ [12]
Menu
Terminal
Terminal
2015-1 192.168.159.128

Wrapping Up

- Signals are a way to interrupt processes.
  - They are not, however, reliable, because:
- Processes can (block) signals.
- A process can only have one signal of a given type pending at a time.