Signals

Posix Signals

- Signals are an integral part of multitasking in the UNIX/POSIX environment. Signals are used for many purposes, including:
  - Exception handling (bad pointer accesses, divide by zero, etc.)
  - Process notification of asynchronous event (I/O completion, timer expiration, etc.)
  - Process termination in abnormal circumstances
  - Interprocess communication

- Signals are similar to the notion of hardware interrupts. However, they are managed and delivered by the Operating System.

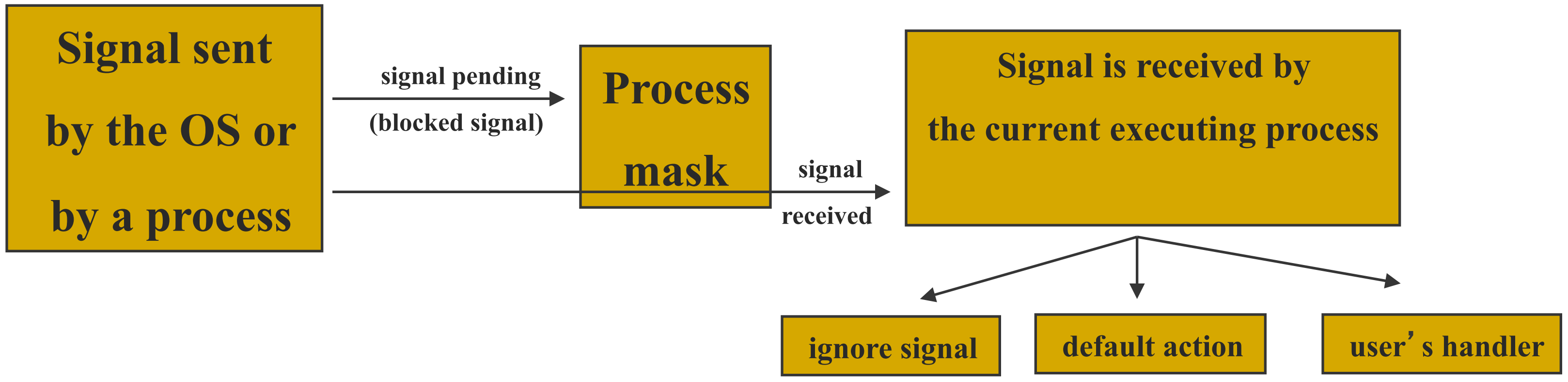
Dealing with signals

- There are different ways in which you can deal with a signal:
  - You can block a signal for a while, and get to it (by unblocking it) later. Blocking signals is a temporary measure.
  - You can ignore the signal, in which case it is as if the signal never arrived.
  - You can handle the signal by executing a default action to deal with the signal (the default action often is to kill the process receiving the signal)
  - You can handle the signal by setting up a function to be called whenever a signal with a particular number arrives.

- There are two spare signals available to user applications: **SIGUSR1** and **SIGUSR2**. Any application can use them as it wants.

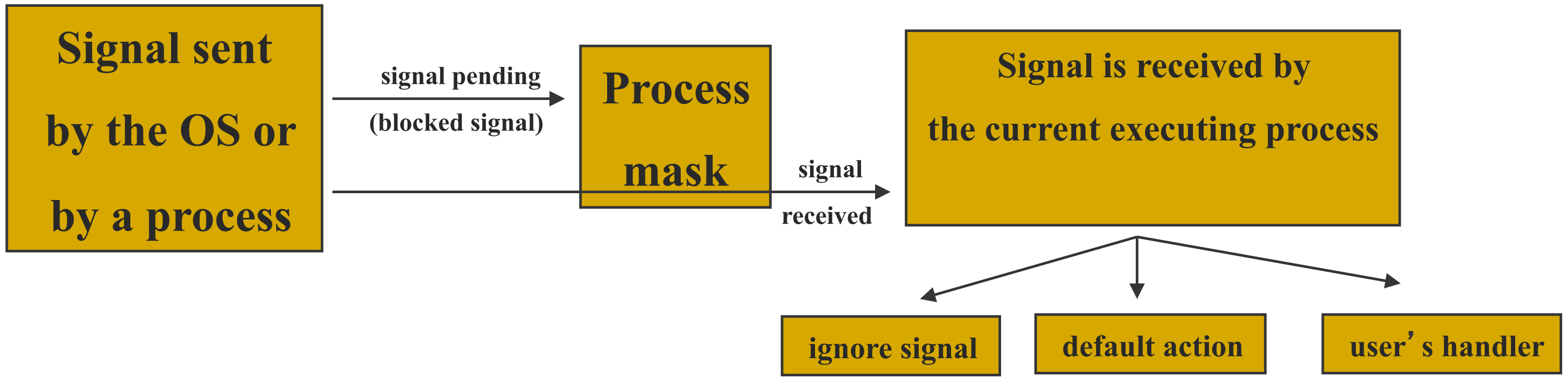
Steps of Signal' s Delivery and Handling

- Event of **sending a signal** to a process:
  - The OS updates the process descriptor to notify that there is a pending signal.
  - At any time, only one pending signal of a given type may exist for a process; additional pending signals of the same type to the same process are not queued but simply discarded (each signal type has a binary flag).

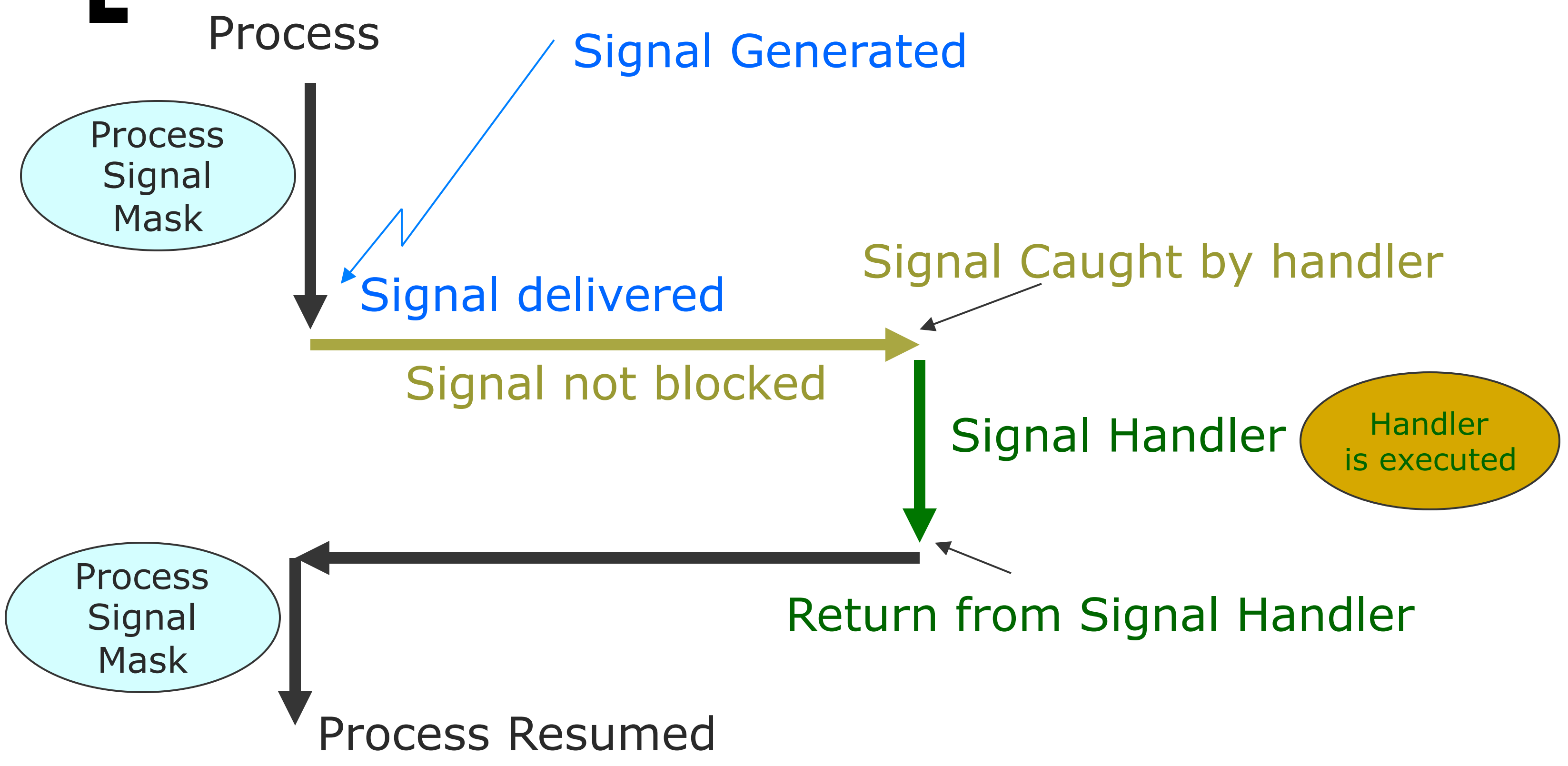


Steps of Signal’s Delivery and Handling

- Event of **receiving a signal**:
  - If the sent signal is blocked by the process mask, the process will not receive the signal until it removes the block: the signal remains pending.
  - If the sent signal is received by the process, the process can ignore the signal, or execute a default action, or execute user’s signal handler.


Signal sent by the OS or by a process
signal pending (blocked signal)
Process mask
signal received
Signal is received by the current executing process
ignore signal
default action
user’s handler

How Signals Work



A signal handler interacts with the regular execution flow of the corresponding process by simply sharing global variables: **the regular execution flow and signal handler share the same address space**.

Examples of POSIX Required Signals

Signal	Description	default action
SIGALRM	**Timer signal**	**Terminate process**
SIGBUS	Bus error (bad memory access)	Terminate process and core dump
SIGCHLD	child terminated or stopped	ignore
SIGINT	**Interrupt from keyboard (usually ctrl-C)**	**Terminate process**
SIGKILL	**Kill signal (cannot be blocked; e.g., kill -9 pid)**	**Terminate process**
SIGUSR1	**User-defined signal 1**	**Terminate process**
SIGUSR2	**User-defined signal 2**	**Terminate process**

Each process uses binary flag for each type of pending signal

```
// Example tested on Linux
#define  N 10
int ccount = 0;

void child_handler(int sig)
{
    pid_t pid = wait(NULL);
    ccount++;
    printf("MSG #%d: Received signal %d from process %d\n",
           ccount, sig, pid);
}

int main()
{
  pid_t pid[N];
  int i, child_status;
  signal(SIGCHLD, child_handler);
  for (i = 0; i < N; i++)
    if ((pid[i] = fork()) == 0) {
      /* Child: Exit */
      exit(0);
    }
  while (1);
}
```

```
Necessary includes:
#include <signal.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
```

Testing the example

Output

```
------------------------------------------
mcaccamo@versilia:~/Dropbox/uiuc/cs241_s14$ ./signal
MSG #1: Received signal 17 from process 13290
MSG #2: Received signal 17 from process 13291
MSG #3: Received signal 17 from process 13292
MSG #4: Received signal 17 from process 13294
MSG #5: Received signal 17 from process 13295
MSG #6: Received signal 17 from process 13296
MSG #7: Received signal 17 from process 13298
MSG #8: Received signal 17 from process 13299
^C
mcaccamo@versilia:~/Dropbox/uiuc/cs241_s14$
```