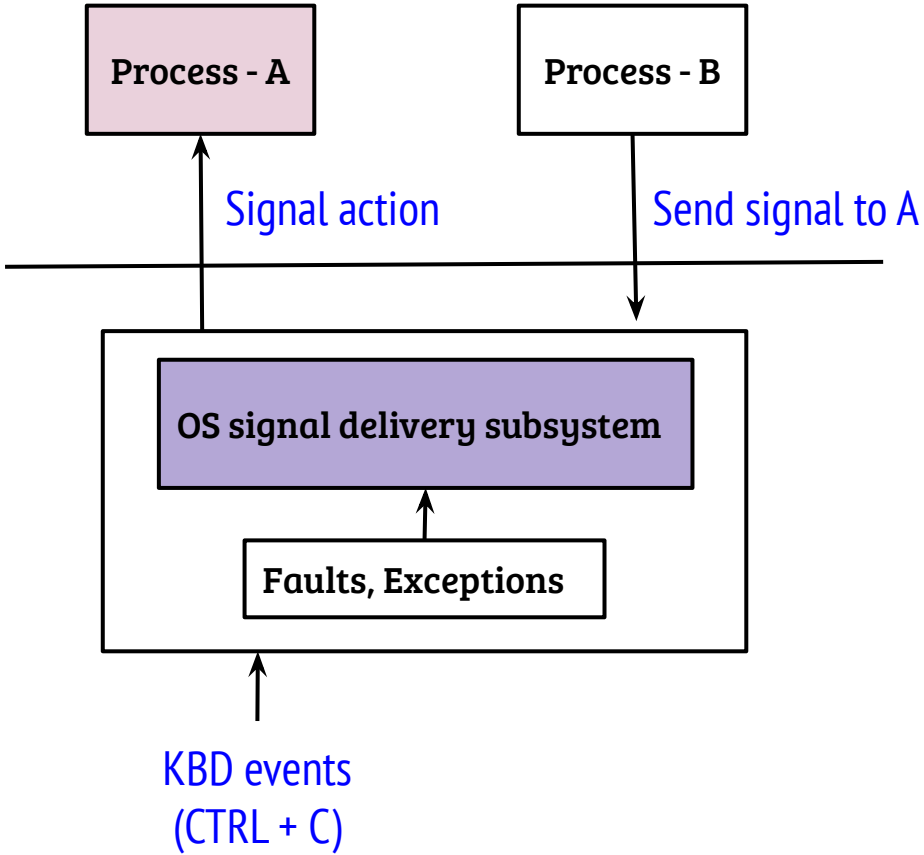
CS330: Operating Systems

Signals

Recap: Process creation

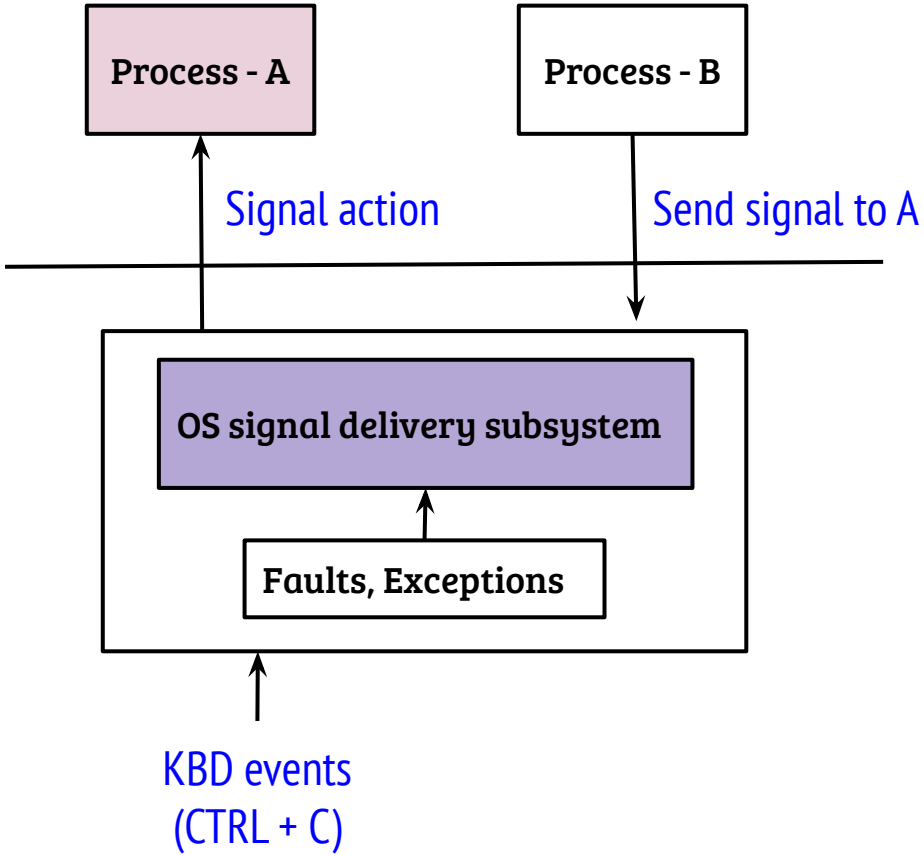
- How does OS come into action after typing "./a.out" in a shell?
- System calls invoked to explicitly give control to the OS
- What exact system calls are invoked?
- fork( ), exec ( ), wait( ) and exit( )
- Who invokes the system calls?
- The shell process (bash process)
- What is the first user process?
- In Unix systems, it is called the *init* process
- Who creates and schedules the init process?
- After boot, the OS creates a PCB and loads the init executable

OS signal mechanism



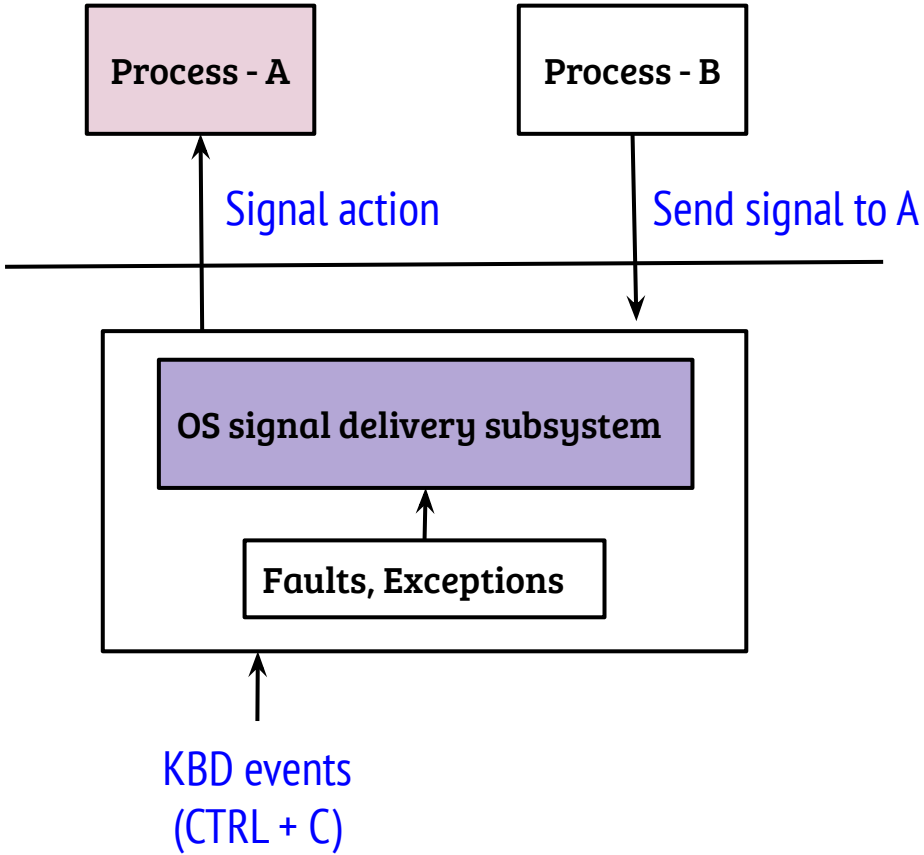
- A signal can be sent from two sources
  - Hardware events
  - Another process
- Step(1): Signal handler is registered (by process-A)

OS signal mechanism



- A signal can be sent from two sources
  - Hardware events
  - Another process
- Step(1): Signal handler is registered (by process-A)
- Step(2): OS logic invoked through syscall or hardware event

OS signal mechanism

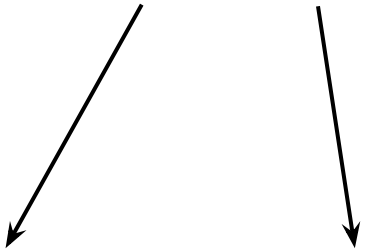


- A signal can be sent from two sources
  - Hardware events
  - Another process
- Step(1): Signal handler is registered (by process-A)
- Step(2): OS logic invoked through syscall or hardware event
- Step(3): Signal handler is invoked

Signal semantics

Destination process

signal (signum, handler)

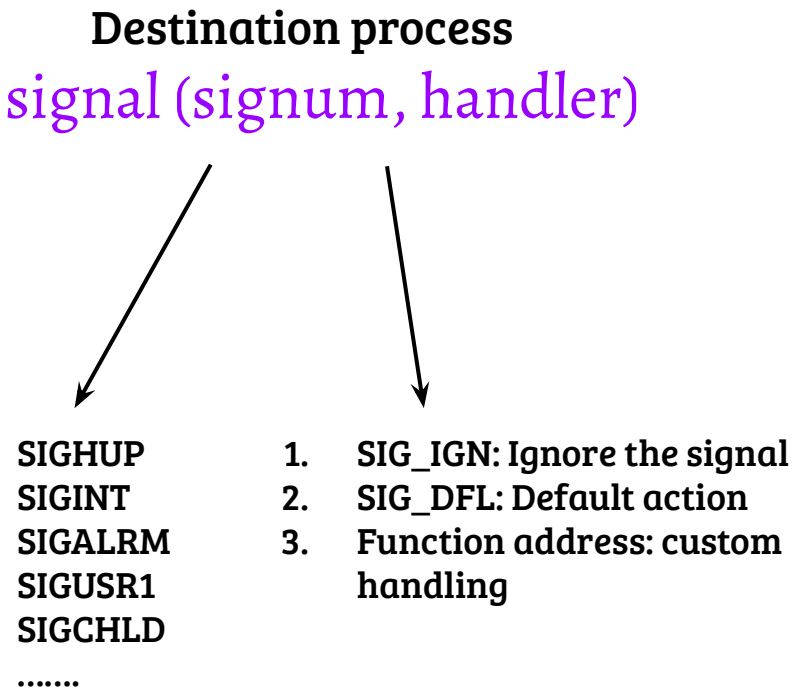


SIGHUP
SIGINT
SIGALRM
SIGUSR1
SIGCHLD
…….

1. **SIG_IGN: Ignore the signal**
2. **SIG_DFL: Default action**
3. **Function address: custom handling**

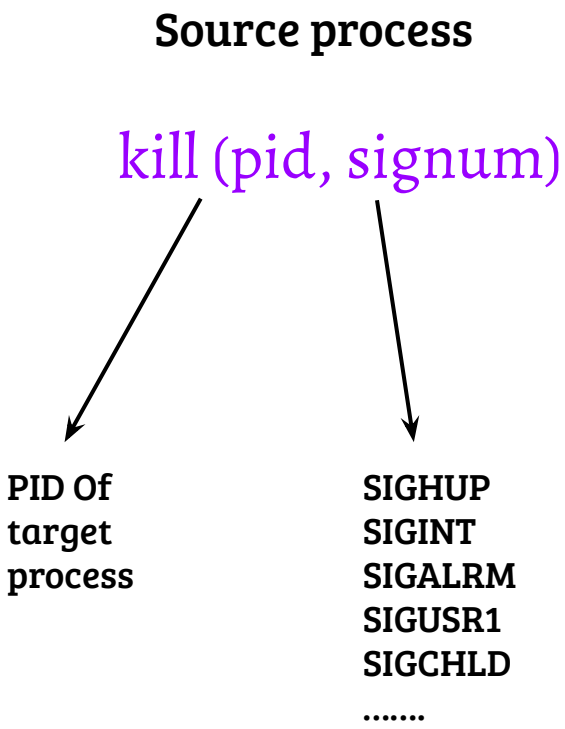
- If signal handler not registered, process is terminated (mostly)
- SIGKILL and SIGSTOP ⟶ no custom actions, why?

Signal semantics



- If signal handler not registered, process is terminated (mostly)
- SIGKILL and SIGSTOP ⟶ no custom actions, why?
- OS or root user should have some way to kill (rouge) processes

Signal semantics

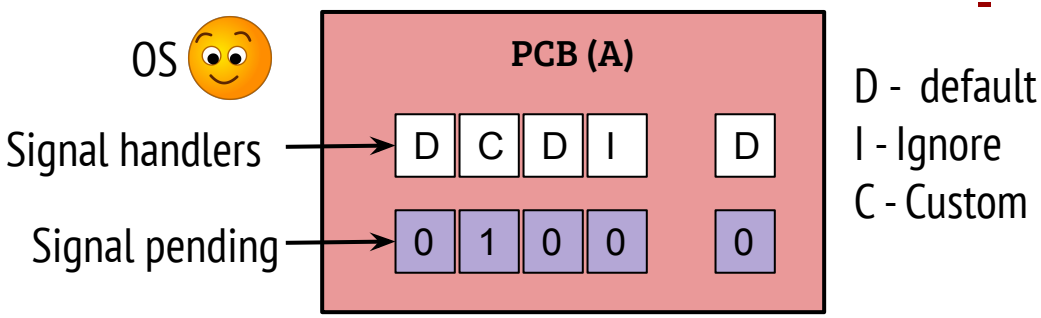


- If pid == 0, signal is sent to all processes in the process group
- Must have permissions to send signals ⟶ same user or root user

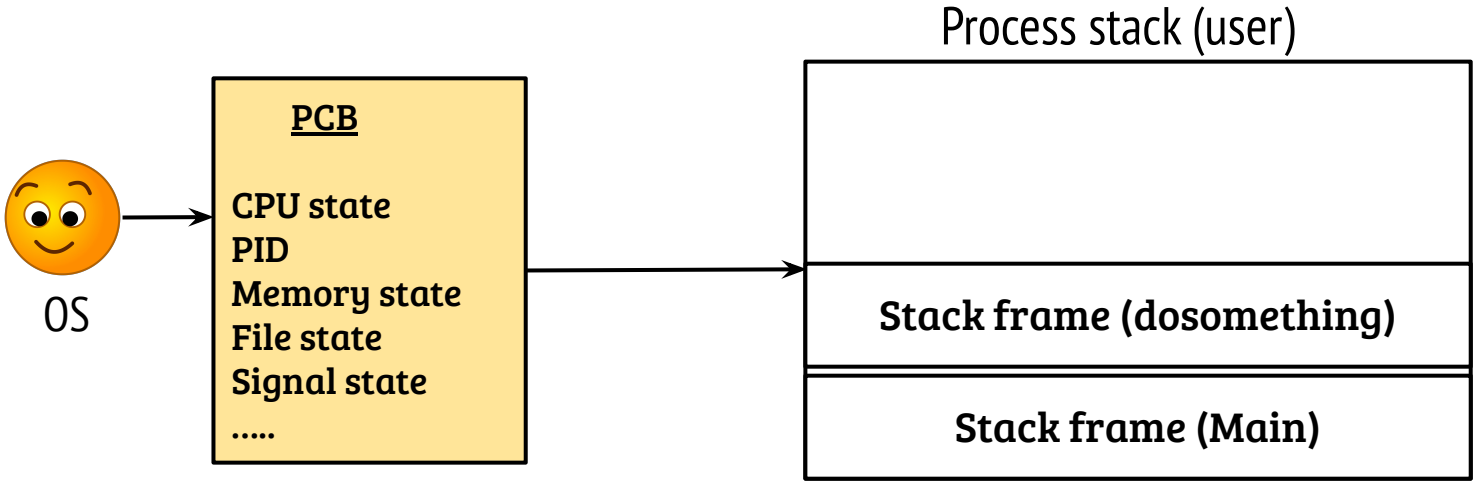
OS support for signals

```
void sighandler(int signo){
   printf("Signal received\n");
}
main( ){
    signal(1, sighandler);
    while(1)  do_something( );
}
```

- Signal handlers are registered by the process
- Signal pending is modified
  - *kill* system call
  - OS event handler
- How are signals delivered?
- When are signals delivered?

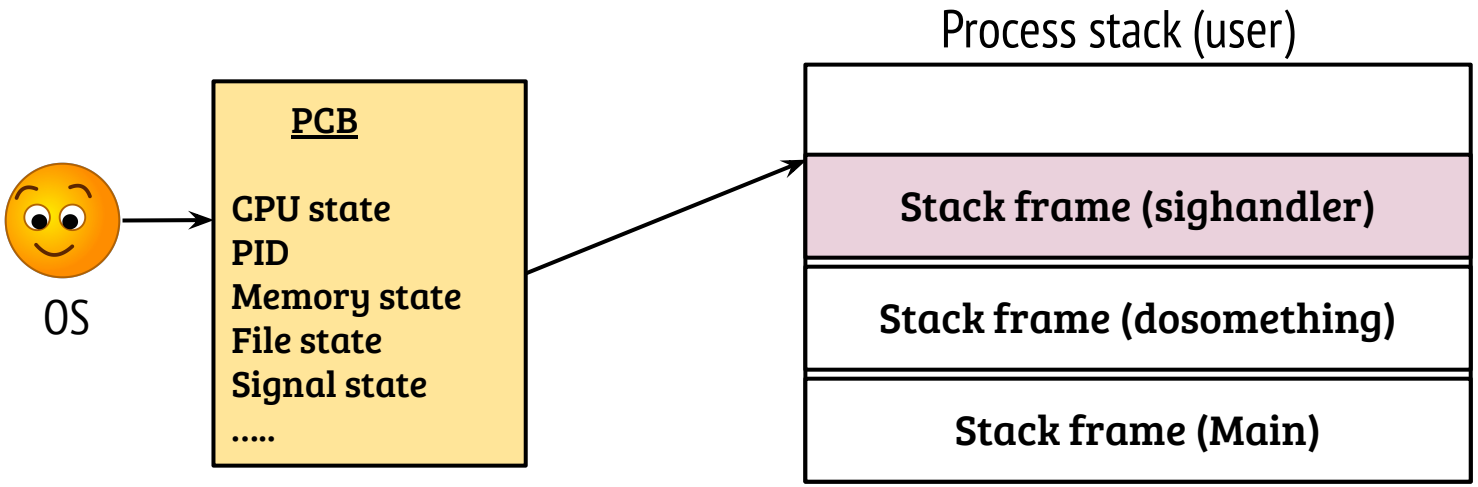


Signal delivery (invoking the handler)



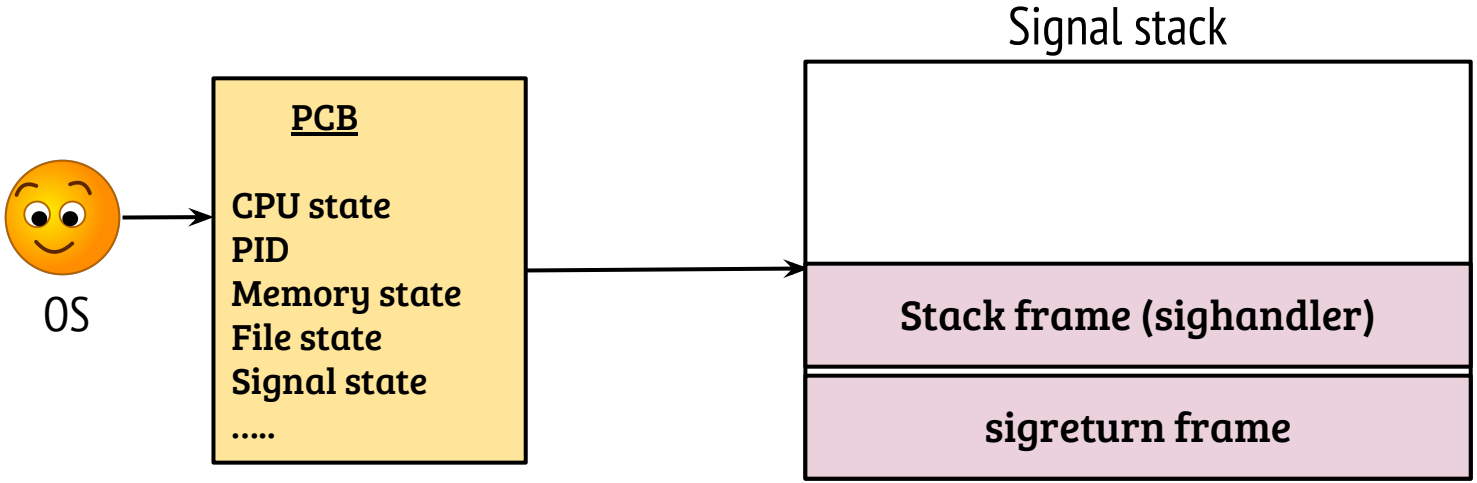
- How to invoke the signal handler?
- Current state: PC points to the last execution address, SP points to the TOS

Signal delivery using the user stack



- How to invoke the signal handler?
- Current state: PC points to the last execution address, SP points to the TOS
- Mimic a function call by modifying user stack and PC → sighandler

Signal delivery using signal stack (Linux)

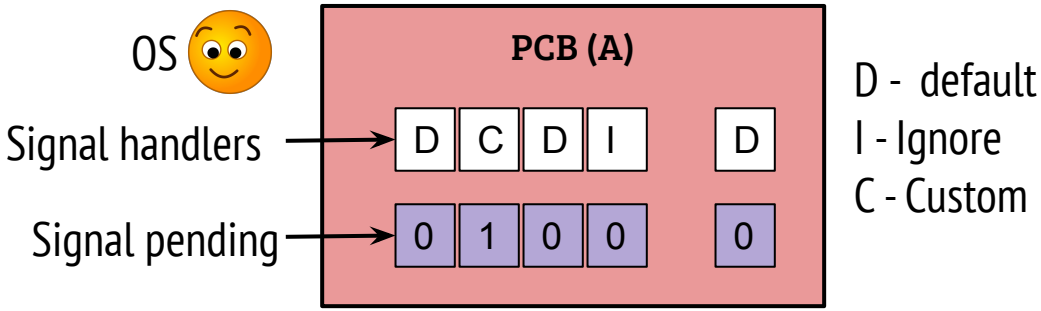


- OS allocates a stack before invoking the handler, *remembers the old stack*
- Creates a stack frame to invoke an *sigreturn* system call to free the stack
- Original stack restored by OS *sigreturn* handler

OS support for signals

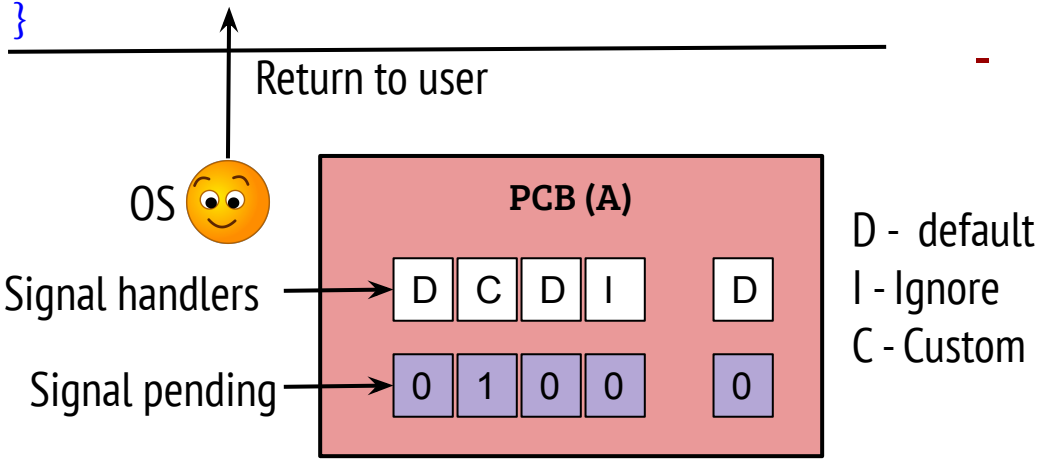
```
void sighandler(int signo){
   printf("Signal received\n");
}
main( ){
    signal(1, sighandler);
    while(1)  do_something( );
}
```

- How are signals delivered?
  - By modifying the user stack and PC address
  - Using an alternate temporary (signal) stack
- When are signals delivered?



OS support for signals

```
void sighandler(int signo){
   printf("Signal received\n");
}
main( ){
    signal(1, sighandler);
    while(1)  do_something( );
}
```



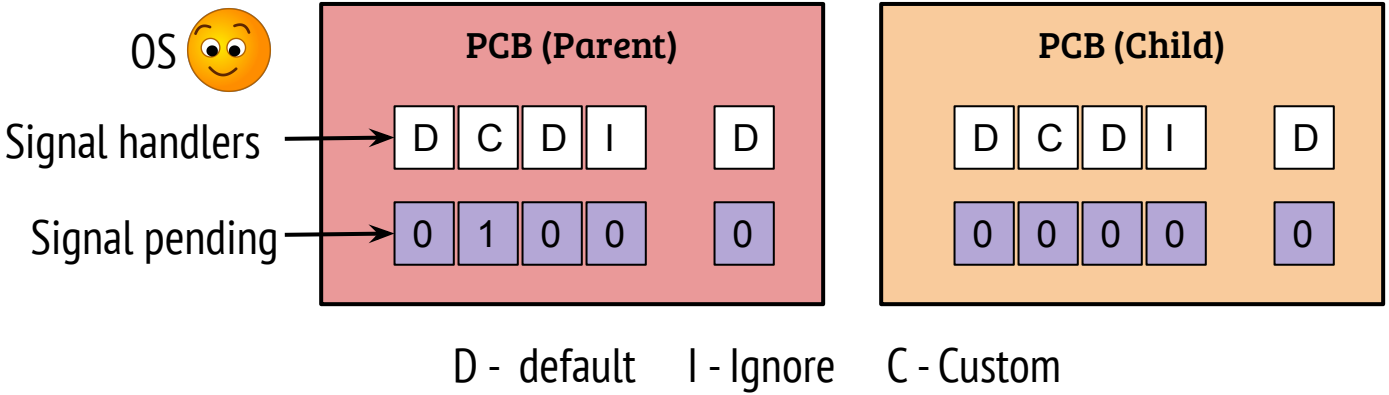
- How are signals delivered?
  - By modifying the user stack and PC address
  - Using an alternate temporary (signal) stack
- When are signals delivered?
  - On return to user space

Signal and fork

```
main( ){
    signal(1, sighandler);
    fork( );
    .....
}
```

USER

- Child inherits the signal handlers
- Note that, signal pending is not copied
- What is the logic?



Signal and fork

```
main( ){
    signal(1, sighandler);
    fork( );
    .....
}
```

- Child inherits the signal handlers
- Note that, signal pending is not copied
- What is the logic?
  - Signal intended for the parent, not for child