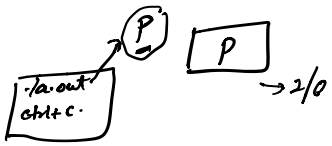
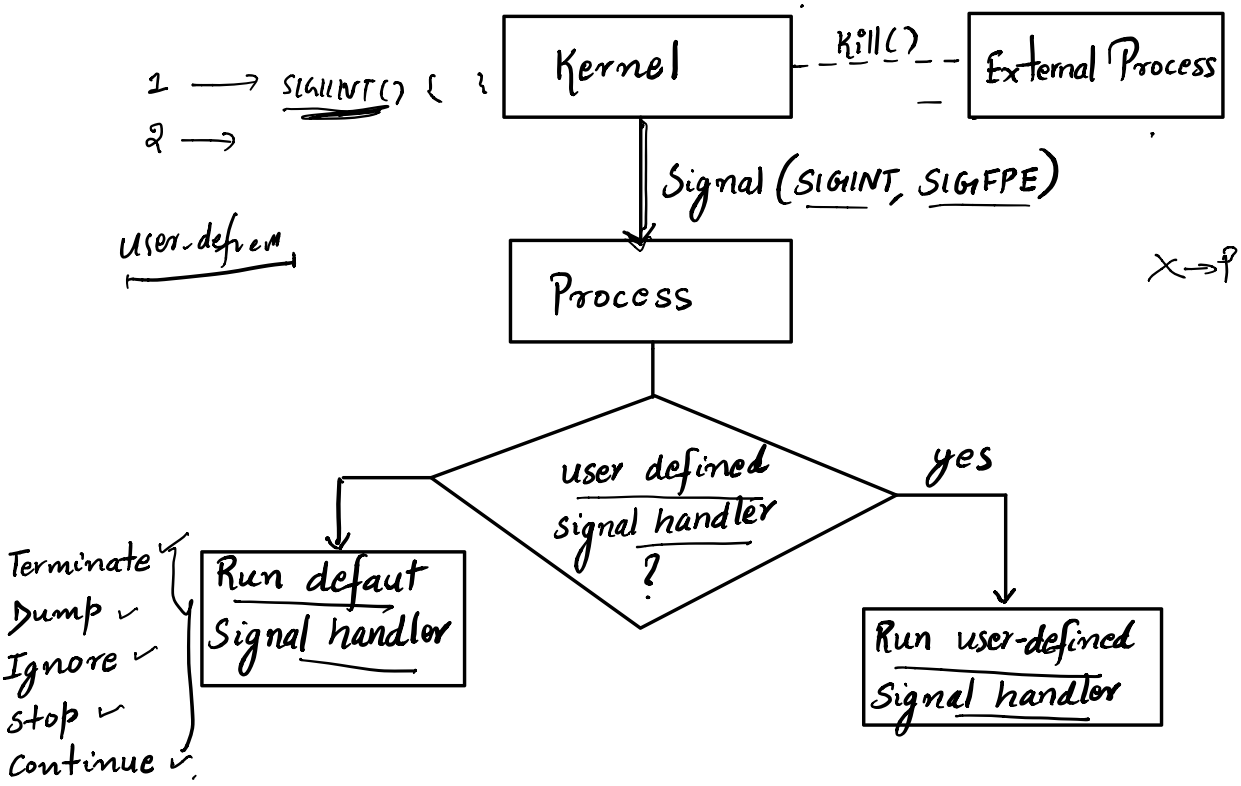
SIGNALS IN OPERATING SYSTEM

- Signals are a limited form of inter-process communication
- A signal is a notification sent to a process in order to notify it of an event that occurred
- Signals can be delivered synchronously or asynchronously
- Division by zero - SIGFPE - synchronous
- ctrl + c - SIGINT - asynchronous

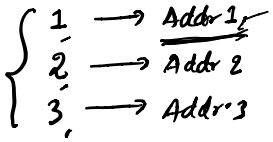


What happens when the signal is sent?

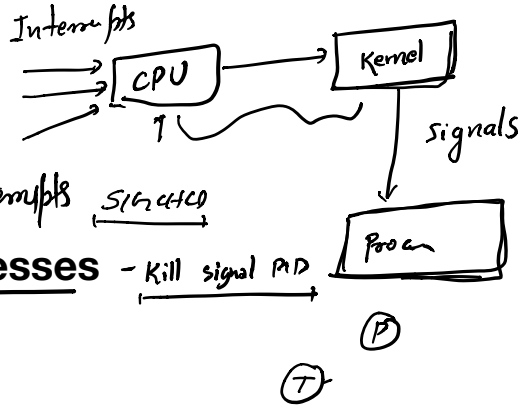
- Signals can be directly send from the kernel or by a process
- They are eventually managed by the kernel who delivers them



INTERRUPTS VS SIGNALS



- Interrupts are initiated by the CPU(divide by zero,page fault,segmentation fault) or CPU instruction(traps) or devices(keyboard,mouse)
- CPU will then interrupt the current task and inform kernel about the interrupt.
- Hardware interrupt - directly run the interrupt handler
- Software interrupts - kernel sends a signal to the process
- Remember, signals are not always sent in response to a software interrupt
- SIGCHLD signal is sent to the parent process when it's child terminates
- SIGINT can be a result of a software interrupt or can be sent by an external process
- Interrupts can be viewed as a communication between the CPU and OS kernel
- Signals can be viewed as a form of communication between OS Kernel and processes



- An event has occurred:
  - An event on CPU - software interrupts SIGCHLD
  - An event from external processes - Kill signal PID
  - A kernel event

Examples of signals in Linux

- ctrl + c - SIGINT - terminates a process
- ctrl + z - SIGSTP - suspends a process from terminal
- SIGSTOP - suspends a process - cannot be ignored
- SIGFPE - divide by zero - terminate
- SIGSEGV - segmentation fault - terminate
- SIGKILL - terminate immediately - cannot be ignored
- SIGCHLD - when child terminates - ignored by default