DEVLINUX
ACADEMY

IPC SIGNAL

“Code Your Embedded Future”

https://devlinux.vn

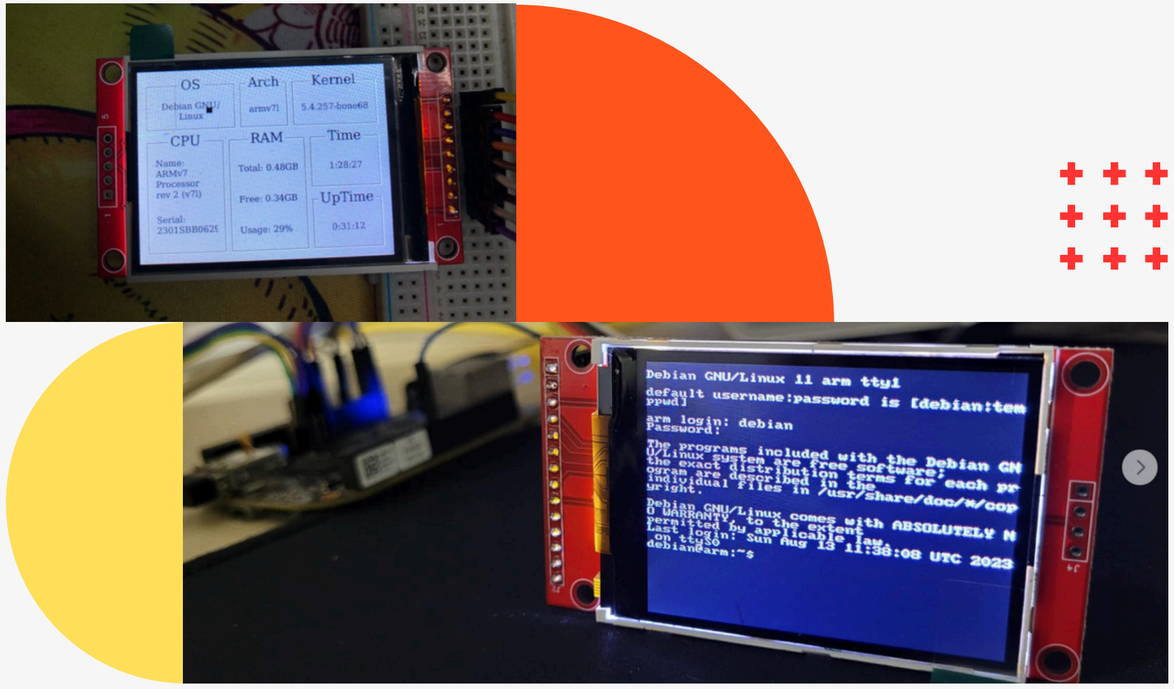
Cộng đồng lập trình Nhúng & Vi Mạch

SUBSCRIBE @devlinux097

DEVLINUX ACADEMY

"Code Your Embedded Future"

AGENDA



- I. Introduction
- II. Signal Handler
- III. Some Common Signals
- IV. Sending Signal
- V. Blocking & Unblocking Signals

https://devlinux.vn

Cộng đồng lập trình Nhúng & Vi Mạch

SUBSCRIBE @devlinux097

DEVLINUX
ACADEMY

INTRODUCTION

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

What is Signal?

Signal is one of the oldest interprocess communication methods in Unix System. Signal is a software interrupt, a mechanism for handling asynchronous events.

Example

These events can originate externally, such as when the user presses Ctrl+C or from operations within the program, such as dividing a number by zero.

Signal Lifecycle

Generation: First a signal is raised/sent/generated.
Delivery: A signal is pending until it is delivered.
Processing: Once the signal is delivered, it can be processed in a variety of ways.

- **Ignore the signal.**
- **Catch and handle the signal.**
- **Perform the default action.**

Processing

Ignore the signal:

- No action is performed.
- SIGKILL and SIGSTOP cannot be ignored.

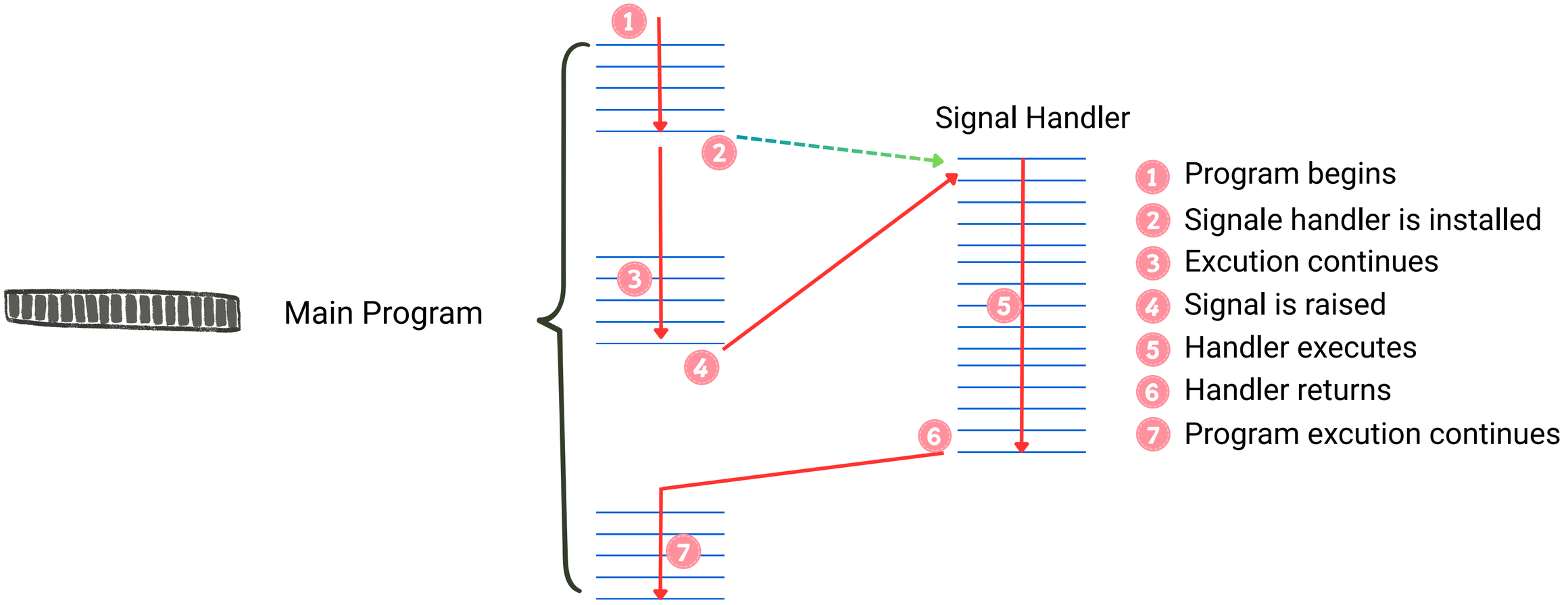
Catch and handle the signal:

- The kernel will pause the main thread and jump to the signal handler registered by the user in the process.
- SIGINT and SIGTERM are two commonly used signals.
- SIGKILL and SIGSTOP cannot be caught.

Perform the default action:

- This action depends on the type of signal.

Signal Lifecycle



DEVLINUX
ACADEMY

SIGNAL HANDLER

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

System call signal()

We register the signal handler via the signal() system call.

Signal is a software interrupt so it is quite sensitive to execution time. When the signal handler is executed it will completely occupy the CPU of the process.

It is necessary to exit the signal handler function as quickly as possible.

```
sighandler_t signal (int signo, sighandler_t handler);
```

We register the handling of a signal
Arguments:

- signo: signal number.
- handler: signal handler.

DEVLINUX
ACADEMY

SOME COMMON SIGNALS

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

Some common signals

SIGKILL : Can only be sent by the kill() system call. Process cannot be caught or ignored. By default, it will terminate the specified process.

SIGTERM : Can only be sent by the kill() system call. By default, it will terminate the specified process, however the process can catch this signal and clean up before terminating.

SIGINT : This signal is sent to processes in the foreground process group. By default, it will terminate the current process.

SIGCHLD : Whenever a process stops, it sends SIGCHLD to its parent process. By default, SIGCHLD is ignored.

SIGSTOP : Can only be sent by the kill() system call. Process cannot be caught or ignored. By default, it will suspend the specified process.

SIGUSR1/SIGUSR2: User-defined signals available.

DEVLINUX
ACADEMY

SENDING SIGNAL

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

System call kill()

Signals can be sent via the kill() system call in the source code.

They can also be sent via the kill command on the terminal.

Signals can be sent to the process itself using the getpid() function.

int kill (pid_t pid, int signo);

Send a signal to a process with the specified pid.
Arguments:

- pid: PID of the process.
- Signo: signal number.

Returns 0 on success, less than 0 on failure.

DEVLINUX
ACADEMY

BLOCKING & UNBLOCKING SIGNAL

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

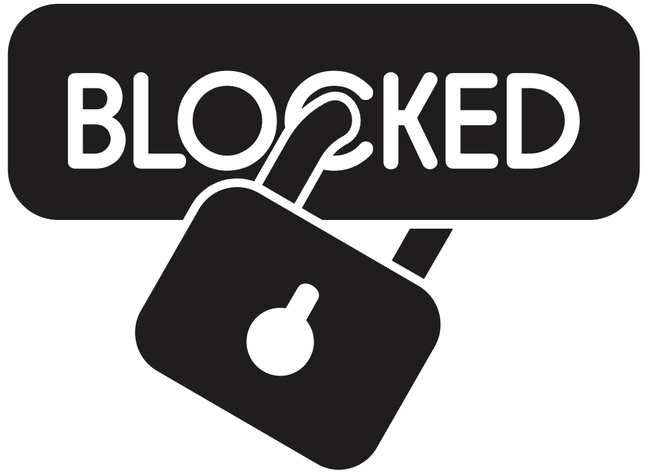
Definition

Signals interrupt the execution of a process. This is often undesirable when the process is executing some important code. Blocking signals solve this problem.

Each process can specify which specific signals it wants to block. If the signal is blocked, it is held in a pending queue by the kernel.

The signal is only sent to the process after it is unblocked.

The list of blocked signals is called the signal mask



Blocking & Unblocking signal

System calls to manipulate signals:

- int sigemptyset (sigset_t *set);
- int sigfillset (sigset_t *set);
- int sigaddset (sigset_t *set, int signo);
- int sigdelset (sigset_t *set, int signo);
- int sigismember (const sigset_t *set, int signo);

Blocking & Unblocking signal

int sigprocmask (int how, const sigset_t *newset, sigset_t *oldset);

how

- SIG_SETMASK: the process's signal mask will be changed to newset.
- SIG_BLOCK: newset will be added to the signal mask (OR).
- SIG_UNBLOCK: newset will be removed from the signal mask

If oldset is not NULL, sigprocmask will extract the current signal mask and store it in oldset.

If newset is NULL, sigprocmask will skip changing the signal mask, but it will extract the current signal mask and store it in oldset. In other words, passing null to set is a way to extract the current signal mask.

DEVLINUX
ACADEMY

Thanks for Your Attention!

Liên hệ với chúng tôi

https://devlinux.vn

Cộng đồng lập trình Nhúng & Vi Mạch

SUBSCRIBE @devlinux097