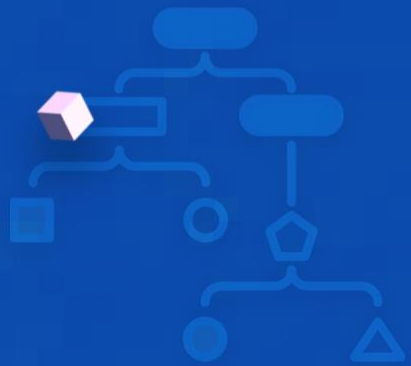
강원혁신플랫폼

리눅스프로그래밍

시그널 응용



Linux Signal에서 정의되지 않은 동작을 일으키거나 프로그램의 정상적인 실행을 방해하지 않고 신호 처리기 내에서 안전하게 호출 할 수 있는 함수를 무엇이라고 하나요?

Signal Safety 함수

학습목표

학습 내용

1. Signal 마스킹
2. 비동기 Signal, Signal Safety
3. Signal Queuing, Signal Driven

학습 목표

- Signal 마스킹에 대해 설명할 수 있다.
- 비동기 Signal, Signal Safety에 대해 설명할 수 있다.
- Signal Queuing, Signal Driven I/O에 대해 설명할 수 있다.

리눅스(Linux) 시그널 응용

Signal 마스킹

Signal 마스킹 및 차단

- Signal 마스크 소개 및 Signal을 일시적으로 차단하는 방법
- 시그널 마스크를 관리하기 위한 sigprocmask() 함수
- 경쟁 조건을 피하기 위한 중요 섹션 및 신호 차단

따라하기 SignalMaskingBlocking.c


```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

// Signal handler function
void sigint_handler(int signum) {
    printf("\nReceived SIGINT signal. Press 'q' to quit.\n");
}

int main() {
    struct sigaction sa;
    sa.sa_handler = sigint_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
```

따라하기 SignalMaskingBlocking.c


```
// Register the signal handler for SIGINT
if (sigaction(SIGINT, &sa, NULL) == -1) {
    perror("Error registering signal handler");
    return 1;
}

// Block SIGINT signal
sigset_t new_mask, old_mask;
sigemptyset(&new_mask);
sigaddset(&new_mask, SIGINT);
if (sigprocmask(SIG_BLOCK, &new_mask, &old_mask) < 0) {
    perror("Error blocking SIGINT");
    return 1;
}
```

```
    printf("SIGINT is blocked. Press 'q' to unblock and quit.\n");

    // Wait for the user to input 'q' to unblock the signal and quit
    char input;
    do {
        input = getchar();
    } while (input != 'q');

    // Restore the old signal mask (unblock SIGINT)
    if (sigprocmask(SIG_SETMASK, &old_mask, NULL) < 0) {
        perror("Error restoring signal mask");
        return 1;
    }
    printf("SIGINT is unblocked. Exiting the program.\n");
    return 0;
}
```

따라하기 SignalMaskingBlocking.c, 실행결과


```
$ gcc -o SignalMaskingBlocking SignalMaskingBlocking.c
$ ./SignalMaskingBlocking
SIGINT is blocked. Press 'q' to unblock and quit.
^C
q

Received SIGINT signal. Press 'q' to quit.
SIGINT is unblocked. Exiting the program.
$
```

비동기 Signal

비동기 Signal 처리

- 비동기 Signal 및 잠재적인 문제를 처리
- Signal 처리기에서 재진입 Signal 안전 기능의 역할

따라하기 AsynchronousSignalHandling.c


```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

// Signal handler function
void sigusr1_handler(int signum) {
    printf("Received SIGUSR1 signal asynchronously.\n");
}

int main() {
    struct sigaction sa;
    sa.sa_handler = sigusr1_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
```

```
    // Register the signal handler for SIGUSR1
    if (sigaction(SIGUSR1, &sa, NULL) == -1) {
        perror("Error registering signal handler");
        return 1;
    }

    printf("Waiting for SIGUSR1 signal. Use 'kill -SIGUSR1 %d' to send the signal.\n", getpid());

    while (1) {
        // Perform some non-signal-related work
        sleep(1);
        printf("Doing some work...\n");
    }

    return 0;
}
```

따라하기AsynchronousSignalHandling.c, 실행결과


```
$ gcc -o AsynchronousSignalHandling AsynchronousSignalHandling.c
$ ./AsynchronousSignalHandling
Waiting for SIGUSR1 signal. Use 'kill -SIGUSR1 1470' to send the signal.
Doing some work...
Doing some work...
Doing some work...
Doing some work...
Doing some work...
Received SIGUSR1 signal asynchronously.
Doing some work...
Received SIGUSR1 signal asynchronously.
Doing some work... $
```

다른 터미널에서 실행

```
$ kill -SIGUSR1 1470
$ kill -SIGUSR1 1470
```

Signal Safety

Signal Safety 사용

- C 프로그램에서 신호 안전 및 재진입을 보장
- Signal 처리기 내에서 Signal 안전 함수만 사용
- 신호 처리기의 문제를 피하기 위해 신호 안전 코드 작성

Signal Safety 함수

- 정의되지 않은 동작을 일으키거나 프로그램의 정상적인 실행을 방해하지 않고 신호 처리기 내에서 안전하게 호출
- 신호 처리기에서 async-signal-safe 함수 사용
- signal-safety(7) — Linux manual page

Function	Notes
abort(3)	Added in POSIX.1-2001 TC1
accept(2)	
access(2)	
alarm(2)	
bind(2)	
chmod(2)	
dup(2)	
execl(3)	Added in POSIX.1-2008;
getpid(2)	
kill(2)	
link(2)	
listen(2)	
open(2)	
pipe(2)	
pthread_kill(3)	Added in POSIX.1-2008 TC1
raise(3)	
read(2)	
select(2)	
signal(2)	
socket(2)	
write(2)	

따라하기 SignalSafetyReentrancy.c


```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void sigusr1_handler(int signum) {
    // Using a signal-safe function (write) to print a message
    const char msg[] = "Received SIGUSR1 signal.\n";
    write(STDOUT_FILENO, msg, sizeof(msg) - 1);
}

int main() {
    struct sigaction sa;
    sa.sa_handler = sigusr1_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
```

따라하기 SignalSafetyReentrancy.c


```
    // Register the signal handler for SIGUSR1
    if (sigaction(SIGUSR1, &sa, NULL) == -1) {
        perror("Error registering signal handler");
        return 1;
    }
    printf("Waiting for SIGUSR1 signal. Use 'kill -SIGUSR1 %d' to send the signal.\n", getpid());

    while (1) {
        // Perform some non-signal-related work
        sleep(1);
        printf("Doing some work...\n");
    }

    return 0;
}
```

따라하기 SignalSafetyReentrancy.c, 실행결과


```
$ gcc -o SignalSafetyReentrancy SignalSafetyReentrancy.c
$ ./SignalSafetyReentrancy
Waiting for SIGUSR1 signal. Use 'kill -SIGUSR1 1581' to send the signal.
Doing some work...
Doing some work...
Doing some work...
Received SIGUSR1 signal.
Doing some work...
Received SIGUSR1 signal.
Received SIGUSR1 signal.
Doing some work...

다른 터미널에서 실행
$ kill -SIGUSR1 1581
$ kill -SIGUSR1 1581
```

Signal Queuing

Signal Queuing 사용

- Signal 대기열을 사용하면 신호와 함께 추가 데이터를 프로세스로 전송
- 신호 이벤트와 관련된 특정 정보를 전달해야 할 때 유용함
- sigqueue() : 신호와 함께 값 또는 포인터를 포함하는 기능 제공

따라하기 SignalQueuing.c


```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

// Signal handler function
void sigusr1_handler(int signum, siginfo_t* info, void* context) {
    printf("Received SIGUSR1 signal with value: %d\n", info->si_value.sival_int);
}

int main() {
    struct sigaction sa;
    sa.sa_flags = SA_SIGINFO; // Use extended signal handling with siginfo_t
    sa.sa_sigaction = sigusr1_handler;
```

```
    // Register the signal handler for SIGUSR1
    if (sigaction(SIGUSR1, &sa, NULL) == -1) {
        perror("Error registering signal handler");
        return 1;
    }

    // Sending SIGUSR1 signal with a value (e.g., 42)
    union sigval value;
    value.sival_int = 42;
    if (sigqueue(getpid(), SIGUSR1, value) == -1) {
        perror("Error sending signal");
        return 1;
    }
    printf("SIGUSR1 signal sent with value: %d\n", value.sival_int);
    return 0;
}
```

따라하기 SignalQueuing.c, 실행결과


```
$ gcc -o SignalQueuing SignalQueuing.c
$ ./SignalQueuing
Received SIGUSR1 signal with value: 42
SIGUSR1 signal sent with value: 42
$
```

요약정리

01 • Signal 마스킹

02 • 비동기 Signal, Signal Safety

03 • Signal Queuing