NAME

setjmp, sigsetjmp, longjmp, siglongjmp – non-local goto

SYNOPSIS

#include <setjmp.h>

int setjmp(jmp_buf *env***);**

int sigsetjmp(sigjmp_buf *env***, int** *savemask***);**

void longjmp(jmp_buf *env***, int** *val***);**

void siglongjmp(sigjmp_buf *env***, int** *val***);**

MT-LEVEL

Unsafe

DESCRIPTION

These functions are useful for dealing with errors and interrupts encountered in a low-level subroutine of a program.

setjmp( ) saves its stack environment in *env* for later use by **longjmp( )**.

sigsetjmp( ) saves the calling process's registers and stack environment (see **sigaltstack**(2)) in *env* for later use by **siglongjmp( )**. If *savemask* is non-zero, the calling process's signal mask (see **sigprocmask**(2)) and scheduling parameters (see **priocntl**(2)) are also saved.

longjmp( ) restores the environment saved by the last call of **setjmp( )** with the corresponding *env* argument. After **longjmp( )** is completed, program execution continues as if the corresponding call of **setjmp( )** had just returned the value *val*. The caller of **setjmp( )** must not have returned in the interim. **longjmp( )** cannot cause **setjmp( )** to return the value 0. If **longjmp( )** is invoked with a second argument of 0, **setjmp( )** will return 1. At the time of the second return from **setjmp( )**, all external and static variables have values as of the time **longjmp( )** is called (see example).

siglongjmp( ) restores the environment saved by the last call of **sigsetjmp( )** with the corresponding *env* argument. After **siglongjmp( )** is completed, program execution continues as if the corresponding call of **sigsetjmp( )** had just returned the value *val*. **siglongjmp( )** cannot cause **sigsetjmp( )** to return the value 0. If **siglongjmp( )** is invoked with a second argument of 0, **sigsetjmp( )** will return 1. At the time of the second return from **sigsetjmp( )**, all external and static variables have values as of the time **siglongjmp( )** is called.

If a signal-catching function interrupts **sleep( )** and calls **siglongjmp( )** to restore an environment saved prior to the **sleep( )** call, the action associated with **SIGALRM** and time it is scheduled to be generated are unspecified. It is also unspecified whether the **SIGALRM** signal is blocked, unless the process's signal mask is restored as part of the environment.

The function **siglongjmp( )** restores the saved signal mask if and only if the *env* argument was initialized by a call to the **sigsetjmp( )** function with a non-zero *savemask* argument.

The values of register and automatic variables are undefined. Register or automatic variables whose value must be relied upon must be declared as **volatile**.

EXAMPLES

The following example uses both **setjmp( )** and **longjmp( )** to return the flow of control to the appropriate instruction block:

```
#include <stdio.h>
#include <setjmp.h>
#include <signal.h>
#include <unistd.h>
jmp_buf env; static void signal_handler();

main() {
    int returned_from_longjump, processing = 1;
    unsigned int time_interval = 4;
    if ((returned_from_longjump = setjmp(env)) != 0)
      switch (returned_from_longjump)    {
       case SIGINT:
        printf("longjumped from interrupt %d\n",SIGINT);
        break;
       case SIGALRM:
        printf("longjumped from alarm %d\n",SIGALRM);
        break;
      }
    (void) signal(SIGINT, signal_handler);
    (void) signal(SIGALRM, signal_handler);
    alarm(time_interval);
    while (processing)      {
     printf(" waiting for you to INTERRUPT (cntrl-C) ...\n");
     sleep(1);
    }                        /* end while forever loop */
}

static void signal_handler(sig)
int sig; {
    switch (sig)    {
     case SIGINT:          ...  /* process for interrupt */
                           longjmp(env,sig);
                            /* break never reached */
     case SIGALRM:         ...  /* process for alarm */
                           longjmp(env,sig);
                            /* break never reached */
     default:              exit(sig);
    }
}
```

When this example is compiled and executed, and the user sends an interrupt signal, the output will be:

longjumped from interrupt

Additionally, every 4 seconds the alarm will expire, signalling this process, and the output will be:

longjumped from alarm

RETURN VALUES

If **longjmp( )** or **siglongjmp( )** are invoked with a second argument of 0, **setjmp( )** and **sigsetjmp( ) ,** respectively, return 1. Otherwise, **setjmp( )** and **sigsetjmp( )** return 0.

SEE ALSO

getcontext(2), **priocntl**(2), **sigaction**(2), **sigaltstack**(2), **sigprocmask**(2), **signal**(3C)

WARNINGS

If **longjmp( )** or **siglongjmp( )** are called even though *env* was never primed by a call to **setjmp( )** or **sigsetjmp( )**, or when the last such call was in a function that has since returned, absolute chaos is guaranteed.