Red Hat Enterprise Linux Release 9.2 Manual Pages on 'getcontext.3' command

$ man getcontext.3

GETCONTEXT(3)   Linux Programmer's Manual   GETCONTEXT(3)

NAME

    getcontext, setcontext - get or set the user context

SYNOPSIS

    #include <ucontext.h>

    int getcontext(ucontext_t *ucp);

    int setcontext(const ucontext_t *ucp);

DESCRIPTION

    In a System V-like environment, one has the two types mcontext_t and

    ucontext_t defined in <ucontext.h> and the four functions getcontext(),

    setcontext(), makecontext(3), and swapcontext(3) that allow user-level

    context switching between multiple threads of control within a process.

    The mcontext_t type is machine-dependent and opaque. The ucontext_t

    type is a structure that has at least the following fields:

        typedef struct ucontext_t {

            struct ucontext_t *uc_link;

            sigset_t   uc_sigmask;

            stack_t   uc_stack;

            mcontext_t   uc_mcontext;

            ...

        } ucontext_t;

    with sigset_t and stack_t defined in <signal.h>. Here uc_link points

    to the context that will be resumed when the current context terminates

(in case the current context was created using makecontext(3)), uc_sig?

mask is the set of signals blocked in this context (see sigproc?

mask(2)), uc_stack is the stack used by this context (see sigalt?

stack(2)), and uc_mcontext is the machine-specific representation of

the saved context, that includes the calling thread's machine regis?

ters.

The function getcontext() initializes the structure pointed to by ucp

to the currently active context.

The function setcontext() restores the user context pointed to by ucp.

A successful call does not return. The context should have been ob?

tained by a call of getcontext(), or makecontext(3), or received as the

third argument to a signal handler (see the discussion of the SA_SIG?

INFO flag in sigaction(2)).

If the context was obtained by a call of getcontext(), program execu?

tion continues as if this call just returned.

If the context was obtained by a call of makecontext(3), program execu?

tion continues by a call to the function func specified as the second

argument of that call to makecontext(3). When the function func re?

turns, we continue with the uc_link member of the structure ucp speci?

fied as the first argument of that call to makecontext(3). When this

member is NULL, the thread exits.

If the context was obtained by a call to a signal handler, then old

standard text says that "program execution continues with the program

instruction following the instruction interrupted by the signal". How?

ever, this sentence was removed in SUSv2, and the present verdict is

"the result is unspecified".

RETURN VALUE

When successful, getcontext() returns 0 and setcontext() does not re?

turn. On error, both return -1 and set errno appropriately.

ERRORS

None defined.

ATTRIBUTES

For an explanation of the terms used in this section, see at?

tributes(7).

```
?????????????????????????????????????????????????????????????
?Interface           ? Attribute   ? Value         ?
?????????????????????????????????????????????????????????????
?getcontext(), setcontext() ? Thread safety ? MT-Safe race:ucp ?
?????????????????????????????????????????????????????????????
```

CONFORMING TO

SUSv2, POSIX.1-2001. POSIX.1-2008 removes the specification of getcon?
text(), citing portability issues, and recommending that applications
be rewritten to use POSIX threads instead.

NOTES

The earliest incarnation of this mechanism was the setjmp(3)/longjmp(3)
mechanism. Since that does not define the handling of the signal con?
text, the next stage was the sigsetjmp(3)/siglongjmp(3) pair. The
present mechanism gives much more control. On the other hand, there is
no easy way to detect whether a return from getcontext() is from the
first call, or via a setcontext() call. The user has to invent their
own bookkeeping device, and a register variable won't do since regis?
ters are restored.

When a signal occurs, the current user context is saved and a new con?
text is created by the kernel for the signal handler. Do not leave the
handler using longjmp(3): it is undefined what would happen with con?
texts. Use siglongjmp(3) or setcontext() instead.

SEE ALSO

sigaction(2), sigaltstack(2), sigprocmask(2), longjmp(3), makecon?
text(3), sigsetjmp(3), signal(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project.   A
description of the project, information about reporting bugs, and the
latest   version   of   this   page,   can   be   found   at
https://www.kernel.org/doc/man-pages/.

Linux   2020-12-21   GETCONTEXT(3)