Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man swapcontext.3

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

NAME

makecontext, swapcontext - manipulate user context

SYNOPSIS

#include <ucontext.h>

void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...);

int swapcontext(ucontext_t *oucp, const ucontext_t *ucp);

DESCRIPTION

In a System V-like environment, one has the type ucontext_t (defined in <ucontext.h> and described in getcontext(3)) and the four functions getcontext(3), setcontext(3), makecontext(), and swapcontext() that al? low user-level context switching between multiple threads of control within a process.

The makecontext() function modifies the context pointed to by ucp (which was obtained from a call to getcontext(3)). Before invoking makecontext(), the caller must allocate a new stack for this context and assign its address to ucp->uc_stack, and define a successor context and assign its address to ucp->uc_link.

When this context is later activated (using setcontext(3) or swapcon? text()) the function func is called, and passed the series of integer (int) arguments that follow argc; the caller must specify the number of these arguments in argc. When this function returns, the successor context is activated. If the successor context pointer is NULL, the

thread exits.

The swapcontext() function saves the current context in the structure pointed to by oucp, and then activates the context pointed to by ucp.

RETURN VALUE

When successful, swapcontext() does not return. (But we may return later, in case oucp is activated, in which case it looks like swapcon? text() returns 0.) On error, swapcontext() returns -1 and sets errno appropriately.

ERRORS

ENOMEM Insufficient stack space left.

VERSIONS

makecontext() and swapcontext() are provided in glibc since version 2.1.

ATTRIBUTES

For an explanation of the terms used in this section, see at? tributes(7).

Interface	Attribute	Value
makecontext()	Thread safety	MT-Safe race:ucp
swapcontext()	Thread safety	MT-Safe race:oucp race:ucp

CONFORMING TO

SUSv2, POSIX.1-2001. POSIX.1-2008 removes the specifications of make? context() and swapcontext(), citing portability issues, and recommend? ing that applications be rewritten to use POSIX threads instead.

NOTES

The interpretation of ucp->uc_stack is just as in sigaltstack(2), namely, this struct contains the start and length of a memory area to be used as the stack, regardless of the direction of growth of the stack. Thus, it is not necessary for the user program to worry about this direction.

On architectures where int and pointer types are the same size (e.g., x86-32, where both types are 32 bits), you may be able to get away with passing pointers as arguments to makecontext() following argc. How? ever, doing this is not guaranteed to be portable, is undefined accord? ing to the standards, and won't work on architectures where pointers are larger than ints. Nevertheless, starting with version 2.8, glibc makes some changes to makecontext(), to permit this on some 64-bit ar? chitectures (e.g., x86-64).

EXAMPLES

The example program below demonstrates the use of getcontext(3), make? context(), and swapcontext(). Running the program produces the follow? ing output:

```
$ ./a.out
main: swapcontext(&uctx_main, &uctx_func2)
func2: started
func2: swapcontext(&uctx_func2, &uctx_func1)
func1: started
func1: swapcontext(&uctx_func1, &uctx_func2)
func2: returning
func1: returning
main: exiting
```

Program source

```
#include <ucontext.h>
#include <stdio.h>
#include <stdlib.h>
static ucontext_t uctx_main, uctx_func1, uctx_func2;
#define handle_error(msg) \
    do { perror(msg); exit(EXIT_FAILURE); } while (0)
static void
func1(void)
{
    printf("func1: started\n");
    printf("func1: swapcontext(&uctx_func1, &uctx_func2)\n");
```

```
    if (swapcontext(&uctx_func1, &uctx_func2) == -1)
        handle_error("swapcontext");
    printf("func1: returning\n");
}
static void
func2(void)
{
    printf("func2: started\n");
    printf("func2: swapcontext(&uctx_func2, &uctx_func1)\n");
    if (swapcontext(&uctx_func2, &uctx_func1) == -1)
        handle_error("swapcontext");
    printf("func2: returning\n");
}
int
main(int argc, char *argv[])
{
    char func1_stack[16384];
    char func2_stack[16384];
    if (getcontext(&uctx_func1) == -1)
        handle_error("getcontext");
    uctx_func1.uc_stack.ss_sp = func1_stack;
    uctx_func1.uc_stack.ss_size = sizeof(func1_stack);
    uctx_func1.uc_link = &uctx_main;
    makecontext(&uctx_func1, func1, 0);
    if (getcontext(&uctx_func2) == -1)
        handle_error("getcontext");
    uctx_func2.uc_stack.ss_sp = func2_stack;
    uctx_func2.uc_stack.ss_size = sizeof(func2_stack);
    /* Successor context is f1(), unless argc > 1 */
    uctx_func2.uc_link = (argc > 1) ? NULL : &uctx_func1;
    makecontext(&uctx_func2, func2, 0);
    printf("main: swapcontext(&uctx_main, &uctx_func2)\n");
    if (swapcontext(&uctx_main, &uctx_func2) == -1)
```

```
            handle_error("swapcontext");
        printf("main: exiting\n");
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

sigaction(2), sigaltstack(2), sigprocmask(2), getcontext(3), sigsetjmp(3)

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/.