NAME

ffi_call - Invoke a foreign function.

SYNOPSIS

#include <ffi.h>

void
ffi_call(*ffi_cif *cif, void (*fn)(void), void *rvalue, void **avalue*);

DESCRIPTION

The **ffi_call** function provides a simple mechanism for invoking a function without requiring knowledge of the function's interface at compile time. *fn* is called with the values retrieved from the pointers in the *avalue* array. The return value from *fn* is placed in storage pointed to by *rvalue*. *cif* contains information describing the data types, sizes and alignments of the arguments to and return value from *fn*, and must be initialized with **ffi_prep_cif** before it is used with **ffi_call**.

rvalue must point to storage that is sizeof(ffi_arg) or larger for non-floating point types. For smaller-sized return value types, the **ffi_arg** or **ffi_sarg** integral type must be used to hold the return value.

EXAMPLES

```
#include <ffi.h>
#include <stdio.h>

unsigned char
foo(unsigned int, float);

int
main(int argc, const char **argv)
{
   ffi_cif cif;
   ffi_type *arg_types[2];
   void *arg_values[2];
   ffi_status status;

   // Because the return value from foo() is smaller than sizeof(long), it
   // must be passed as ffi_arg or ffi_sarg.
   ffi_arg result;

   // Specify the data type of each argument. Available types are defined
   // in <ffi/ffi.h>.
```

```
    arg_types[0] = &ffi_type_uint;
    arg_types[1] = &ffi_type_float;

    // Prepare the ffi_cif structure.
    if ((status = ffi_prep_cif(&cif, FFI_DEFAULT_ABI,
       2, &ffi_type_uint8, arg_types)) != FFI_OK)
    {
       // Handle the ffi_status error.
    }

    // Specify the values of each argument.
    unsigned int arg1 = 42;
    float arg2 = 5.1;

    arg_values[0] = &arg1;
    arg_values[1] = &arg2;

    // Invoke the function.
    ffi_call(&cif, FFI_FN(foo), &result, arg_values);

    // The ffi_arg 'result' now contains the unsigned char returned from foo(),
    // which can be accessed by a typecast.
    printf("result is %hhu", (unsigned char)result);

    return 0;
}

// The target function.
unsigned char
foo(unsigned int x, float y)
{
    unsigned char result = x - y;
    return result;
}
```

SEE ALSO

ffi(3), ffi_prep_cif(3)