## NAME

**stdarg** - variable argument lists

## SYNOPSIS

**#include <stdarg.h>**

*void*
**va_start**(*va_list ap*, *last*);

*type*
**va_arg**(*va_list ap*, *type*);

*void*
**va_copy**(*va_list dest*, *va_list src*);

*void*
**va_end**(*va_list ap*);

## DESCRIPTION

A function may be called with a varying number of arguments of varying types. The include file <*stdarg.h*> declares a type (*va_list*) and defines four macros for stepping through a list of arguments whose number and types are not known to the called function.

The called function must declare an object of type *va_list* which is used by the macros **va_start**(), **va_arg**(), **va_copy**(), and **va_end**().

The **va_start**() macro initializes *ap* for subsequent use by **va_arg**(), **va_copy**(), and **va_end**(), and must be called first.

The parameter *last* is the name of the last parameter before the variable argument list, i.e., the last parameter of which the calling function knows the type.

Because the address of this parameter is used in the **va_start**() macro, it should not be declared as a register variable, or as a function or an array type.

The **va_arg**() macro expands to an expression that has the type and value of the next argument in the call. The parameter *ap* is the *va_list ap* initialized by **va_start**() or **va_copy**(). Each call to **va_arg**() modifies *ap* so that the next call returns the next argument. The parameter *type* is a type name specified so that the type of a pointer to an object that has the specified type can be obtained simply by adding a * to *type*.

If there is no next argument, or if *type* is not compatible with the type of the actual next argument (as promoted according to the default argument promotions), random errors will occur.

The first use of the **va_arg**() macro after that of the **va_start**() macro returns the argument after *last*. Successive invocations return the values of the remaining arguments.

The **va_copy**() macro copies a variable argument list, previously initialized by **va_start**(), from *src* to *dest*. The state is preserved such that it is equivalent to calling **va_start**() with the same second argument used with *src*, and calling **va_arg**() the same number of times as called with *src*.

The **va_end**() macro cleans up any state associated with the variable argument list *ap*.

Each invocation of **va_start**() or **va_copy**() must be paired with a corresponding invocation of **va_end**() in the same function.

## RETURN VALUES

The **va_arg**() macro returns the value of the next argument.

The **va_start**(), **va_copy**(), and **va_end**() macros return no value.

## EXAMPLES

The function *foo* takes a string of format characters and prints out the argument associated with each format character based on the type.

```
void foo(char *fmt, ...)
{
        va_list ap;
        int d;
        char c, *s;

        va_start(ap, fmt);
        while (*fmt)
                switch(*fmt++) {
                case 's':                       /* string */
                        s = va_arg(ap, char *);
                        printf("string %s\n", s);
                        break;
                case 'd':                       /* int */
                        d = va_arg(ap, int);
                        printf("int %d\n", d);
```

```
                        break;
                case 'c':                       /* char */
                        /* Note: char is promoted to int. */
                        c = va_arg(ap, int);
                        printf("char %c\n", c);
                        break;
                }
        va_end(ap);
}
```

## COMPATIBILITY

These macros are *not* compatible with the historic macros they replace. A backward compatible version can be found in the include file <*varargs.h*>.

## STANDARDS

The **va_start**(), **va_arg**(), **va_copy**(), and **va_end**() macros conform to ISO/IEC 9899:1999 ("ISO C99").

## HISTORY

The **va_start**(), **va_arg**() and **va_end**() macros were introduced in ANSI X3.159-1989 ("ANSI C89"). The **va_copy**() macro was introduced in ISO/IEC 9899:1999 ("ISO C99").

## BUGS

Unlike the *varargs* macros, the **stdarg** macros do not permit programmers to code a function with no fixed arguments. This problem generates work mainly when converting *varargs* code to **stdarg** code, but it also creates difficulties for variadic functions that wish to pass all of their arguments on to a function that takes a *va_list* argument, such as vfprintf(3).