III. Ctext Device Interface Functions

NAME

Functions - ctext device interface functions

FUNCTION

Almost all of the ctext system is portable, not only across different implementations of Idris, but across the full range of operating systems where Whitesmiths C is supported. The only dependencies on an underlying operating system are in the device driver programs, and these dependencies, in turn, have been isolated into two areas.

One system dependency involves how a terminal line can be made seven or eight-bit transparent. The other dependency is how (and if) a mechanism can be constructed to time I/O, providing a timeout capability if an operation is not complete within a specified time limit.

The descriptions in this section are portable specifications of the functions used by the device drivers to provide these capabilities. Any implementation of ctext should provide functions compatible with these descriptions, and (hopefully) providing the functionality described.

NAME

mktran - make or unmake transparent data transfer connection

SYNOPSIS

```
COUNT mktran(fd, mode)
    FILE fd;
    COUNT mode;
```

FUNCTION

mktran modifies the file associated with fd, presumably designating a terminal, to enable or disable the transparent transfer of data through the file. If (mode == 1), mktran sets up 7-bit transparent transfer of data; if (2 <= mode), mktran sets up 8-bit transparent transfer of data. If a preceding call to mktran was made with a mode of 1 or 2, then a call with (mode == 0) will restore the terminal characteristics in effect before the first such call; otherwise, a call with (mode == 0) will set up a "normal" (i.e., interpreted) terminal state, which, depending on the underlying operating system, may not be 7-bit transparent.

If (mode < 0), mktran does not change the terminal state; such a call simply reports the current state.

RETURNS

mktran normally returns the mode value (0, 1, or 2) best corresponding to the state of fd before the current call. mktran returns -1 if unable to read or (when required) write the state of fd.

BUGS

There is necessarily some variation in what mktran does under different operating systems; in general, it changes as little as it can to assure the selected transparent connection.

NAME

tmcall - name function to be called on timer interrupt

SYNOPSIS

```
BOOL tmcall(pfn)
    VOID (*pfn)();
```

FUNCTION

tmcall tries to assure that the function whose address is passed in pfn will be called whenever a timer interrupt occurs. (The function tminit initiates the timer; unless tmclr is called before the period specified has elapsed, a timer interrupt will be generated.)

If (pfn == NULL), tmcall will cause timer interrupts to be ignored. If (pfn == 1), tmcall will cause the handling of timer interrupts to revert to the system default. If a timer interrupt occurs before tmcall is called, this default handling will also be in effect.

RETURNS

tmcall returns YES if the handling implied by pfn was successfully associated with the timer interrupt, or NO if any errors occurred.

SEE ALSO

tmclr, tminit

BUGS

On some systems, this routine (and its kin) may be dummies.

NAME
tmclr - clear timing

SYNOPSIS
```
BOOL tmclr()
```

FUNCTION
tmclr clears any pending timer interrupts, by causing the timer to be halted. If no timing is being performed, tmclr does nothing.

RETURNS
tmclr returns YES if timing has been halted, NO if any errors occured in trying to do so.

BUGS
On some systems, tmclr (and its kin) may be dummies.

NAME

tminit - initiate timing

SYNOPSIS

```
BOOL tminit(delay)
    COUNT delay;
```

FUNCTION

tminit attempts to cause a timer interrupt to be generated after the number of seconds specified by delay. Presumably, this interrupt will cause control to be transferred to the routine already named to tmcall.

Only one timer may be in use at any given time; that is, either a timer interrupt or a call to tmclr must occur between each call to tminit.

RETURNS

tminit returns YES if timing was successfully initiated, or NO if any errors prevented it.

SEE ALSO

tmcall, tmclr

BUGS

On some systems, tminit (and its kin) may be dummies.

NAME

tmwait - wait for period given

SYNOPSIS

```
BOOL tmwait(delay)
    COUNT delay;
```

FUNCTION

tmwait attempts to wait for the number of seconds specified by delay. No interrupt of any kind occurs at the end of this period; tmwait simply returns to its caller.

RETURNS

tmwait returns YES if the wait occurred, NO otherwise.

BUGS

On some systems, tmwait may be a dummy.