NAME

dial – establish an outgoing terminal line connection

SYNOPSIS

cc [*flag* ...] *file* ... **–lnsl** [*library* ...]

#include <dial.h>

int dial(CALL *call***);**

void undial(int *fd***);**

MT-LEVEL

Unsafe

DESCRIPTION

dial( ) returns a file-descriptor for a terminal line open for read/write. The argument to **dial( )** is a **CALL** structure (defined in the header **<dial.h>**).

When finished with the terminal line, the calling program must invoke **undial( )** to release the semaphore that has been set during the allocation of the terminal device.

CALL is defined in the header **<dial.h>** and has the following members:

```
struct termio   *attr;      /* pointer to termio attribute struct */
int             baud;       /* transmission data rate */
int             speed;      /* 212A modem: low=300, high=1200 */
char            *line;      /* device name for out-going line */
char            *telno;     /* pointer to tel-no digits string */
int             modem;      /* specify modem control for direct lines */
char            *device;    /* unused */
int             dev_len;    /* unused */
```

The **CALL** element **speed** is intended only for use with an outgoing dialed call, in which case its value should be the desired transmission baud rate. The **CALL** element **baud** is no longer used.

If the desired terminal line is a direct line, a string pointer to its device-name should be placed in the **line** element in the **CALL** structure. Legal values for such terminal device names are kept in the **Devices** file. In this case, the value of the **baud** element should be set to -1. This value will cause **dial** to determine the correct value from the **<Devices>** file.

The **telno** element is for a pointer to a character string representing the telephone number to be dialed. Such numbers may consist only of these characters:

0-9	dial 0-9
*	dial *
#	dial #
=	wait for secondary dial tone
–	delay for approximately 4 seconds

The **CALL** element **modem** is used to specify modem control for direct lines. This element should be non-zero if modem control is required. The **CALL** element **attr** is a pointer to a **termio** structure, as defined in the header **<termio.h>**. A **NULL** value for this pointer element may be passed to the **dial** function, but if such a structure is included, the elements specified in it will be set for the outgoing terminal line before the connection is established. This setting is often important for certain attributes such as parity and baud-rate.

The **CALL** elements **device** and **dev_len** are no longer used. They are retained in the **CALL** structure for compatibility reasons.

RETURN VALUES

On failure, a negative value indicating the reason for the failure will be returned. Mnemonics for these negative indices as listed here are defined in the header **<dial.h>**.

INTRPT	**–1**	/* interrupt occurred */
D_HUNG	**–2**	/* dialer hung (no return from write) */
NO_ANS	**–3**	/* no answer within 10 seconds */
ILL_BD	**–4**	/* illegal baud-rate */
A_PROB	**–5**	/* acu problem (open( ) failure) */
L_PROB	**–6**	/* line problem (open( ) failure) */
NO_Ldv	**–7**	/* can't open Devices file */
DV_NT_A	**–8**	/* requested device not available */
DV_NT_K	**–9**	/* requested device not known */
NO_BD_A	**–10**	/* no device available at requested baud */
NO_BD_K	**–11**	/* no device known at requested baud */
DV_NT_E	**–12**	/* requested speed does not match */
BAD_SYS	**–13**	/* system not in Systems file*/

FILES

/etc/uucp/Devices
/etc/uucp/Systems
/var/spool/uucp/LCK..*tty-device*

SEE ALSO

uucp(1C), **alarm**(2), **read**(2), **write**(2), **termio**(7I)

NOTES

Including the header **<dial.h>** automatically includes the header **<termio.h>**.

An **alarm**(2) system call for 3600 seconds is made (and caught) within the **dial** module for the purpose of ''touching'' the **LCK..** file and constitutes the device allocation semaphore for the terminal device. Otherwise, **uucp**(1C) may simply delete the **LCK..** entry on its 90-minute clean-up rounds. The alarm may go off while the user program is in a **read**(2) or **write**(2) function, causing an apparent error return. If the user program expects to be around for an hour or more, error returns from **read( )**s should be checked for **(errno==EINTR)**, and the **read( )** possibly reissued.

This interface is unsafe in multithreaded applications. Unsafe interfaces should be called only from the main thread.