NAME

errfile – error-log file format

DESCRIPTION

When hardware errors are detected by the system, an error record is generated and passed to the error-logging daemon for recording in the error log for later analysis. The default error log is **/errlog/errfile.**

The format of an error record depends on the type of error that was encountered. Every record, however, has a header with the following format:

```
struct errhdr {
        int             e_type;         /* record type */
        int             e_len;          /* bytes in record (inc hdr) */
        time_t          e_time;         /* time of day */
};
```

The permissible record types are as follows:

```
#define E_GOTS  010             /* Start for UNIX/TS */
#define E_GORT  011             /* Start for UNIX/RT */
#define E_STOP  012             /* Stop */
#define E_TCHG  013             /* Time change */
#define E_CCHG  014             /* Configuration change */
#define E_BLK   020             /* Block device error */
#define E_STRAY 030             /* Stray interrupt */
#define E_PRTY  031             /* Memory parity */
#define E_OVFL  040             /* Software table overflow */
#define E_PRDEV 041             /* File system error */
#defint E_POWER 042             /* Power-fail restart */
```

Some records in the error file are of an administrative nature. These include the startup record that is entered into the file when logging is activated, the stop record that is written if the daemon is terminated "gracefully", and the time-change record that is used to account for changes in the system's time-of-day. These records have the following formats:

```
struct estart {
        struct errhdr   e_hdr;          /* record header */
        int             e_cpu;          /* cpu type */
        int             e_mmr3;         /* contents mem mgmt reg 3 */
        long            e_syssize;      /* 11/70 system memory size */
        int             e_bconf;        /* block dev configuration */
};

struct eend {
        struct errhdr   e_hdr;          /* record header */
        int             e_werr;         /* number of daemon write errors */
};

struct etimchg {
        struct errhdr   e_hdr;          /* record header */
        time_t          e_ntime;        /* new time */
};
```

Stray interrupts cause a record with the following format to be logged in the file:

```
struct estray {
        struct errhdr   e_hdr;          /* record header */
        physadr         e_saddr;        /* stray loc or device addr */
        int             e_sbacty;       /* active block devices */
};
```

Memory subsystem error on 11/70 processors cause the following record to be generated:

```
struct eparity {
        struct errhdr   e_hdr;          /* record header */
        int             e_parreg[4];    /* memory subsys registers */
};
```

Error records for block devices have the following format:

```
struct eblock {
        struct errhdr   e_hdr;          /* record header */
        dev_t           e_dev;          /* "true" major + minor dev no */
        physadr         e_regloc;       /* controller address */
        int             e_bacty;        /* other block I/O activity */
        struct iostat {
          long          io_ops;         /* number read/writes */
          long          io_misc;        /* number "other" operations */
          unsigned      io_unlog;       /* number unlogged errors */
        }               e_stats;
        int             e_bflags;       /* read/write, error, etc */
        int             e_cyloff;       /* logical dev start cyl */
        daddr_t         e_bnum;         /* logical block number */
        unsigned        e_bytes;        /* number bytes to transfer */
        long            e_memadd;       /* buffer memory address */
        unsigned        e_rtry;         /* number retries */
        int             e_nreg;         /* number device registers */
};
```

The following values are used in the flags word:

```
#define E_WRITE 0       /* Write operation */
#define E_READ  1       /* Read operation */
#define E_NOIO  02      /* No I/O pending */
#define E_PHYS  04      /* Physical I/O */
#define E_MAP   010     /* Unibus map in use */
#define E_ERROR 020     /* I/O failed */
```

The "true" major device numbers that identify the failing device are as follows:

```
#define RK0     0
#define RP0     1
#define RF0     2
#define TM0     3
#define TC0     4
#define HP0     5
#define HT0     6
#define HS0     7
#define RLO     8
```

File system soft errors generate records of the following format:

```
struct eprdev {
        struct errhdr   e_hdr;          /* record header */
        short           e_missed;       /* errors not logged since preceding record */
        dev_t           e_fsdev;        /* device with filesystem in error */
        short           e_fserr;        /* type of error */
};
```

Values for e_fserr include:

```
#define E_FSBB  0               /* Bad block */
#define E_FSBC  1               /* Bad count */
#define E_FSNS  2               /* No space */
#define E_FSOI  3               /* Out of inodes */
```

Table overflow errors generate records of the following format:

```
struct eovfl {
        struct errhdr   e_hdr;          /* record header */
        short           e_missed;       /* errors not logged since preceeding record */
        short           e_tabt;         /* type of error */
};
```

Values for e_tabt are:

```
#define E_FILEO  0              /* File table overflow */
#define E_PROCO  1              /* Process table overflow */
#define E_INODEO2               /* Inode table overflow */
#define E_TEXTO  3              /* Text table overflow */
```

Powerfail − restart records have the format:

```
struct e_power {
        struct errhdr   e_hdr;          /* record header */
};
```

SEE ALSO

errdemon(1M)