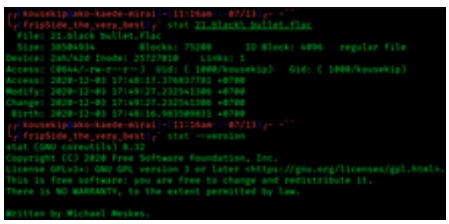
สถิติ (เรียกระบบ)

สถิติ ()เป็นระบบปฏิบัติการยูนิกซ์ เรียกระบบที่ผลตอบแทนคุณลักษณะของแฟ้มเกี่ยวกับinode ความหมายของสถิติ ()แตกต่างกันระหว่างระบบปฏิบัติการ ตัวอย่างเช่นคำสั่ง Unix `ls`ใช้การเรียกระบบนี้เพื่อดึงข้อมูลเกี่ยวกับไฟล์ที่ประกอบด้วย:

- เวลา: เวลาที่เข้าถึงล่าสุด (`ls -lu`)
- mtime: เวลาที่แก้ไขล่าสุด (`ls -l`)
- ctime: เวลาที่เปลี่ยนสถานะล่าสุด (`ls -lc`)



`stat` *บรรทัดคำสั่ง*

`stat`ปรากฏใน1 รุ่นใช้ระบบปฏิบัติการยูนิกซ์ เป็นหนึ่งในไม่กี่การเรียกระบบ Unix ดั้งเดิมที่จะเปลี่ยนแปลงด้วยการเพิ่มการอนุญาตกลุ่มของเวอร์ชัน 4และขนาดไฟล์ที่ใหญ่ขึ้น [1]

ฟังก์ชัน stat()

The C POSIX library header `sys/stat.h`, found on POSIX and other Unix-like operating systems, declares the `stat()` functions, as well as related functions called `fstat()` and `lstat()`. The functions take a `struct stat` buffer argument, which is used to

return the file attributes. On success, the functions return zero, and on error, −1 is returned and errno is set appropriately.

The `stat()` and `lstat()` functions take a filename argument. If the file is a symbolic link, `stat()` returns attributes of the eventual target of the link, while `lstat()` returns attributes of the link itself. The `fstat()` function takes a file descriptor argument instead, and returns attributes of the file that it identifies.

The family of functions was extended to implement large file support. Functions named `stat64()`, `lstat64()` and `fstat64()` return attributes in a `struct stat64` structure, which represents file sizes with a 64-bit type, allowing the functions to work on files 2 GiB and larger (up to 8 EiB). When the `_FILE_OFFSET_BITS` macro is defined to 64, these 64-bit functions are available under the original names.

The functions are defined as:

```
int stat(const char *filename, struct stat *buf);
int lstat(const char *filename, struct stat *buf);
int fstat(int filedesc, struct stat *buf);
```

โครงสร้างสถิติ

This structure is defined in `sys/stat.h` header file as follows, although implementations are free to define additional fields:[2]

```
struct stat {
    mode_t          st_mode;
    ino_t           st_ino;
    dev_t           st_dev;
    dev_t           st_rdev;
    nlink_t         st_nlink;
    uid_t           st_uid;
    gid_t           st_gid;
    off_t           st_size;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    blksize_t       st_blksize;
```

```
    blkcnt_t        st_blocks;
};
```

POSIX.1 does not require `st_rdev`, `st_blocks` and `st_blksize` members; these fields are defined as part of XSI option in the Single Unix Specification.

In older versions of POSIX.1 standard, the time-related fields were defined as `st_atime`, `st_mtime` and `st_ctime`, and were of type `time_t`. Since the 2008 version of the standard, these fields were renamed to `st_atim`, `st_mtim` and `st_ctim`, respectively, of type struct `timespec`, since this structure provides a higher resolution time unit. For the sake of compatibility, implementations can define the old names in terms of the `tv_sec` member of `struct timespec`. For example, `st_atime` can be defined as `st_atim.tv_sec`.[2]

The `struct stat` structure includes at least the following members:

- `st_dev` – identifier of device containing file
- `st_ino` – inode number
- `st_mode` – protection mode; see also Unix permissions
- `st_nlink` – reference count of hard links
- `st_uid` – user identifier of owner
- `st_gid` – group identifier of owner
- `st_rdev` – device identifier (if special file)
- `st_size` – total file size, in bytes
- `st_atime` – time of last access
- `st_mtime` – time of last modification
- `st_ctime` – time of last status change
- `st_blksize` – preferred block size for file system I/O, which can depend upon both the system and the type of file system[3]
- `st_blocks` – number of blocks allocated in multiples of `DEV_BSIZE` (usually 512 bytes).

The `st_mode` field is a bit field. It combines the file access modes and also indicates any special file type. There are many macros to work with the different mode flags and file types.

คำติชมของ atime

Reading a file changes its `atime` eventually requiring a disk *write,* which has been criticized as it is inconsistent with a read only file system. File system cache may significantly reduce this activity to one disk write per cache flush.

Linux kernel developer Ingo Molnár publicly criticized the concept and performance impact of atime in 2007,[4][5] and in 2009, the `relatime` mount option had become the default, which addresses this criticism.[6] The behavior behind the `relatime` mount option offers sufficient performance for most purposes and should not break any significant applications, as it has been extensively discussed.[7] Initially, `relatime` only updated atime if atime < mtime or atime < ctime; that was subsequently modified to update atimes that were 24 hours old or older, so that `tmpwatch` and Debian's popularity counter (popcon) would behave properly.[8]

Current versions of the Linux kernel support four mount options, which can be specified in fstab:

- `strictatime` (formerly `atime`, and formerly the default; `strictatime` as of 2.6.30) – always update atime, which conforms to the behavior defined by POSIX
- `relatime` ("relative atime", introduced in 2.6.20 and the default as of 2.6.30) – only update atime under certain circumstances: if the previous atime is older than the mtime or ctime, or the previous atime is over 24 hours in the past
- `nodiratime` – never update atime of directories, but do update atime of other files
- `noatime` – never update atime of any file or directory; implies `nodiratime`; highest performance, but least compatible
- `lazytime` – update atime according to specific circumstances laid out below

Current versions of Linux, macOS, Solaris, FreeBSD, and NetBSD support a `noatime` mount option in /etc/fstab, which causes the atime field never to be updated. Turning off atime updating breaks POSIX compliance, and some applications, such as mbox-driven "new mail" notifications,[9] and some file usage watching utilities, notably tmpwatch.

The `noatime` option on OpenBSD behaves more like Linux `relatime`.[10]

Version 4.0 of the Linux kernel mainline, which was released on April 12, 2015, introduced the new mount option `lazytime`. It allows POSIX-style atime updates to be performed in-memory and flushed to disk together with some non-time-related I/O operations on the same file; atime updates are also flushed to disk when some of the sync system calls are executed, or before the file's in-memory inode is evicted from the filesystem cache. Additionally, it is

possible to configure for how long atime modifications can remain unflushed. That way, lazytime retains POSIX compatibility while offering performance improvements.[11][12]

ctime

It is tempting to believe that `ctime` originally meant creation time;[13] however, while early Unix did have modification and creation times, the latter was changed to be access time before there was any C structure in which to call anything `ctime`. The file systems retained just the access time (`atime`) and modification time (`mtime`) through 6th edition Unix. The `ctime` timestamp was added in the file system restructuring that occurred with 7th edition Unix, and has always referred to inode change time. It is updated any time file metadata stored in the inode changes, such as file permissions, file ownership, and creation and deletion of hard links. In some implementations, `ctime` is affected by renaming a file: Both original Unix, which implemented a renaming by making a link (updating `ctime`) and then unlinking the old name (updating `ctime` again) and modern Linux tend to do this.

Unlike `atime` and `mtime`, `ctime` cannot be set to an arbitrary value with `utime()`, as used by the `touch` utility, for example. Instead, when `utime()` is used, or for any other change to the inode other than an update to `atime` caused by accessing the file, the `ctime` value is set to the current time.

ความละเอียดของเวลา

- `time_t` provides times accurate to one second.
- Some filesystems provide finer granularity. Solaris 2.1 introduced a microsecond resolution with UFS in 1992 and a nanosecond resolution with ZFS.
- In Linux kernels 2.5.48 and above, the stat structure supports nanosecond resolution for the three file timestamp fields. These are exposed as additional fields in the stat structure.[14][15]
- The resolution of create time on FAT filesystem is 10 milliseconds, while resolution of its write time is two seconds, and access time has a resolution of one day thus it acts as the access date.[16]

ตัวอย่าง

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include <sys/types.h>
#include <pwd.h>
#include <grp.h>
#include <sys/stat.h>

int
main(int argc, char *argv[])
{

    struct stat sb;
    struct passwd *pwuser;
    struct group *grpnam;

    if (argc < 2)
    {
        fprintf(stderr, "Usage: %s: file ...\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    for (int i = 1; i < argc; i++)
    {
        if (-1 == stat(argv[i], &sb))
        {
            perror("stat()");
            exit(EXIT_FAILURE);
        }

        if (NULL == (pwuser = getpwuid(sb.st_uid)))
        {
            perror("getpwuid()");
            exit(EXIT_FAILURE);
        }

        if (NULL == (grpnam = getgrgid(sb.st_gid)))
```

```
        {
            perror("getgrgid()");
            exit(EXIT_FAILURE);
        }

        printf("%s:\n", argv[i]);
        printf("\tinode: %u\n", sb.st_ino);
        printf("\towner: %u (%s)\n", sb.st_uid, pwuser-
>pw_name);
        printf("\tgroup: %u (%s)\n", sb.st_gid, grpnam-
>gr_name);
        printf("\tperms: %o\n", sb.st_mode & (S_IRWXU | S_IRWXG
| S_IRWXO));
        printf("\tlinks: %d\n", sb.st_nlink);
        printf("\tsize: %ld\n", sb.st_size); /* you may use %lld
*/
        printf("\tatime: %s", ctime(&sb.st_atim.tv_sec));
        printf("\tmtime: %s", ctime(&sb.st_mtim.tv_sec));
        printf("\tctime: %s", ctime(&sb.st_ctim.tv_sec));

        printf("\n");
    }

    return 0;
}
```

อ้างอิง

1. *McIlroy, M. D. (1987). A Research Unix reader: annotated excerpts from the Programmer's Manual, 1971–1986 (http://www.cs.dartmouth.edu/~doug/reader.pdf) (PDF) (Technical report). CSTR. Bell Labs. 139.*
2. *Stevens & Rago 2013, p. 94.*
3. *"<sys/stat.h>" (http://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/stat.h.html) . The Open Group Base Specifications Issue 6—IEEE Std 1003.1, 2004 Edition. The Open Group. 2004.*
4. *Kernel Trap: Linux: Replacing atime With relatime (http://kerneltrap.org/node/14148) , by Jeremy, August 7, 2007*

5. *Once upon atime (https://lwn.net/Articles/244829/)* , LWN, by Jonathan Corbet, August 8, 2007
6. *Linux kernel 2.6.30 (http://kernelnewbies.org/Linux_2_6_30)* , Linux Kernel Newbies
7. *That massive filesystem thread (https://lwn.net/Articles/326471/)* , LWN, by Jonathan Corbet, March 31, 2009
8. *Relatime Recap (http://valerieaurora.wordpress.com/2009/03/27/relatime-recap/)* , Valerie Aurora
9. *http://www.mail-archive.com/mutt-users@mutt.org/msg24912.html* "the shell's $MAIL monitor ... depends on atime, pronouncing new email with atime($MAIL) < mtime($MAIL)"
10. *"mount(2) - OpenBSD manual pages" (https://man.openbsd.org/mount.2#MNT_NOATIME)* . openbsd.org. April 27, 2018. Retrieved September 26, 2018.
11. *"Linux kernel 4.0, Section 1.5. 'lazytime' option for better update of file timestamps" (http://kernelnewbies.org/Linux_4.0#head-3e847edbcf4c617048c905b6972979f7bb7547a3)* . kernelnewbies.org. May 1, 2015. Retrieved May 2, 2015.
12. Jonathan Corbet (November 19, 2014). *"Introducing lazytime" (https://lwn.net/Articles/621046/)* . LWN.net. Retrieved May 2, 2015.
13. *"BSTJ version of C.ACM Unix paper" (https://www.bell-labs.com/usr/dmr/www/cacm.html)* .
14. *"stat(2) - Linux manual page" (http://man7.org/linux/man-pages/man2/stat.2.html)* . man7.org. Retrieved February 27, 2015.
15. Andreas Jaeger (December 2, 2002), *struct stat.h with nanosecond resolution (http://www.sourceware.org/ml/libc-alpha/2002-12/msg00011.html)* , mail archive of the libc-alpha@sources.redhat.com mailing list for the glibc project.
16. *MSDN: File Times (http://msdn.microsoft.com/en-us/library/windows/desktop/ms724290%28v=vs.85%29.aspx)*

- IEEE Std 1003.1, 2004, documentation for fstat(2) (http://www.opengroup.org/onlinepubs/009695399/functions/fstat.html) . Retrieved 2012-06-07.
- stat(2) Linux man page (http://linux.die.net/man/2/stat) . Retrieved 2012-06-07.
- W. Richard, Stevens; Stephen A., Rago (May 24, 2013). *Advanced Programming in the UNIX Environment* (http://www.kohala.com/start/apue.html) (Third ed.). Addison-Wesley Professional. ISBN 978-0321637734. Retrieved February 27, 2015.

ลิงค์ภายนอก

- An example showing how to use stat() (http://www.hep.wisc.edu/~pinghc/NoteFileSystemInfo.htm)
- stat() in Perl (http://perldoc.perl.org/functions/stat.html)

- stat() in PHP (http://www.php.net/manual/en/function.stat.php)
- atime and relatime (http://kerneltrap.org/node/14148)
