Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'readahead.2'

$ man readahead.2

readahead(2) System Calls Manual readahead(2)

NAME

readahead - initiate file readahead into page cache

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#define _GNU_SOURCE         /* See feature_test_macros(7) */
#define _FILE_OFFSET_BITS 64
#include <fcntl.h>
ssize_t readahead(int fd, off_t offset, size_t count);
```

DESCRIPTION

readahead() initiates readahead on a file so that subsequent reads from that file will be satisfied from the cache, and not block on disk I/O (assuming the readahead was initiated early enough and that other activity on the system did not in the meantime flush pages from the cache).

The fd argument is a file descriptor identifying the file which is to be read. The offset argument specifies the starting point from which data is to be read and count specifies the number of bytes to be read. I/O is performed in whole pages, so that offset is effectively rounded down to a page boundary and bytes are read up to the next page boundary greater than or equal to (offset+count). readahead() does not read beyond the end of the file. The file offset of the open file description referred to by the file descriptor fd is left unchanged.

RETURN VALUE

On success, readahead() returns 0; on failure, -1 is returned, and errno is set to indicate

the error.

ERRORS

EBADF fd is not a valid file descriptor or is not open for reading.

EINVAL fd does not refer to a file type to which readahead() can be applied.

VERSIONS

On some 32-bit architectures, the calling signature for this system call differs, for the reasons described in syscall(2).

STANDARDS

Linux.

HISTORY

Linux 2.4.13, glibc 2.3.

NOTES

_FILE_OFFSET_BITS should be defined to be 64 in code that uses a pointer to readahead, if the code is intended to be portable to traditional 32-bit x86 and ARM platforms where off_t's width defaults to 32 bits.

BUGS

readahead() attempts to schedule the reads in the background and return immediately. However, it may block while it reads the filesystem metadata needed to locate the requested blocks. This occurs frequently with ext[234] on large files using indirect blocks instead of extents, giving the appearance that the call blocks until the requested data has been read.

SEE ALSO

lseek(2), madvise(2), mmap(2), posix_fadvise(2), read(2)