

## [2.6.18 PATCH]: Filesystem Event Reporter V4 -- test program

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-09/msg09088.html>

---

- *From:* Yi Yang <yang.y.yi@xxxxxxxxxx>
  - *Date:* Sat, 30 Sep 2006 23:45:43 +0800
- 

This is the test program for Filesystem Event Reporter V4, You need to use the header files of built 2.6.18 kernel to compile. Assume your kernel dir is linux-2.6.18, this test program has the same parent directory with linux-2.6.18, then you can compile it using the following command:

```
# gcc -o fsevent-test -I linux-2.6.18/include fsevent-test.c
```

after insmod linux-2.6.18/fs/fsevent.ko, you can run it:

```
# ./fsevent-test > /tmp/fsevent.log
```

you can start some test tools in another terminal, for example ltp, build kernel, fs stress tools, then you view /tmp/fsevent.log to see if it catch those file system operations. If you want to stop it, just press CTRL+C. you can grep system log by the following command to know how many process migration happens.

```
# cat /var/log/messages | grep migration | wc -l
```

if you don't want to use it, rmmmod it to see if it is ok.

```
# rmmmod fsevent
```

Any feedback is welcomed.

Here is fsevent-test.c

```
////////////////////////////////////
```

```
/*
```

```
* fsevent_test.c - Filesystem Events Reporter Userspace Test Program
```

```
* created by Yi Yang <yang.y.yi@xxxxxxxxxx>
```

```
*/
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <sys/types.h>
```

## [2.6.18 PATCH]: Filesystem Event Reporter V4 -- test program

```
#include <sys/socket.h>
#include <signal.h>
#include <linux/netlink.h>
#define _LINUX_TIME_H
#include <linux/fsevent.h>

#define MAX_MSGSIZE 1024
#ifndef SOL_NETLINK
#define SOL_NETLINK 270
#endif
#define MATCH_FSEVENT_TYPE(t1, t2) (((t1) & (t2)) == (t2))

int sd;
struct sockaddr_nl l_local, daddr;
int on;
int len;
struct nlmsg_hdr *nlhdr = NULL;
struct msg_hdr msg;
struct iovec iov;
struct fsevent_filter *filter;
struct cn_msg *cnmsg;
struct fsevent *fsevent;
int counter = 0;
int ret;
struct sigaction sigint_action;
enum fsevent_type type;

void set_fsevent_filter(unsigned int mask, int control, int type, int id)
{
    memset(nlhdr, 0, sizeof(NLMSG_SPACE(MAX_MSGSIZE)));
    memset(&iov, 0, sizeof(struct iovec));
    memset(&msg, 0, sizeof(struct msg_hdr));
    filter = (struct fsevent_filter *)NLMSG_DATA(nlhdr);
    filter->mask = mask;
    if (type != FSEVENT_FILTER_ALL) {
        filter->id.pid = id;
    }
    filter->type = type;
    filter->control = control;
    nlhdr->nlmsg_len = NLMSG_LENGTH(sizeof(struct fsevent_filter));
    nlhdr->nlmsg_pid = getpid();
    nlhdr->nlmsg_flags = 0;
    nlhdr->nlmsg_type = NLMSG_DONE;
    nlhdr->nlmsg_seq = 0;

    iov.iov_base = (void *)nlhdr;
    iov.iov_len = nlhdr->nlmsg_len;
    msg.msg_name = (void *)&daddr;
    msg.msg_namelen = sizeof(daddr);
    msg.msg_iov = &iov;
    msg.msg_iovlen = 1;
}
```

## [2.6.18 PATCH]: Filesystem Event Reporter V4 -- test program

```
ret = sendmsg(sd, &msg, 0);
if (ret == -1) {
perror("sendmsg error:");
exit(-1);
}
}
int get_fsevent()
{
memset(nlhdr, 0, NLMSG_SPACE(MAX_MSGSIZE));
memset(&iov, 0, sizeof(struct iovec));
memset(&msg, 0, sizeof(struct msghdr));

iovec.iov_base = (void *)nlhdr;
iovec.iov_len = NLMSG_SPACE(MAX_MSGSIZE);
msg.msg_name = (void *)&daddr;
msg.msg_namelen = sizeof(daddr);
msg.msg_iov = &iov;
msg.msg_iovlen = 1;

return recvmsg(sd, &msg, 0);
}
void stop_listen()
{
set_fsevent_filter(0xffffffff, FSEVENT_FILTER_IGNORE,
FSEVENT_FILTER_ALL,0);
set_fsevent_filter(0xffffffff, FSEVENT_FILTER_REMOVE,
FSEVENT_FILTER_ALL,0);
}
void sigint_handler(int signo)
{
stop_listen();
printf("filesystem event: turn off filesystem event listening.\n");
get_fsevent();
get_fsevent();
close(sd);
exit(0);
}
void print_fsevent(struct fsevent * event, char * typestr, int flag)
{
printf("filesystem event: %s\n"
"process '%s' %s %s %s",
typestr,
fsevent->name,
typestr,
((fsevent->type & FSEVENT_ISDIR)
== FSEVENT_ISDIR)?"dir":"file",
fsevent->name + fsevent->pname_len + 1);
if (flag)
printf(" to '%s'", fsevent->name + fsevent->pname_len
+ fsevent->fname_len + 2);
printf("\n");
}
```

## [2.6.18 PATCH]: Filesystem Event Reporter V4 -- test program

```
}
void print_ack(char * typestr)
{
printf("filesystem event: "
"acknowledge for filter on %s\n", typestr);
}
int main(void)
{
memset(&sigint_action, 0, sizeof(struct sigaction));
sigint_action.sa_flags = SA_ONESHOT;
sigint_action.sa_handler = &sigint_handler;
sigaction(SIGINT, &sigint_action, NULL);
nlhdr = (struct nlmsgshdr *)malloc(NLMSG_SPACE(MAX_MSGSIZE));
if (nlhdr == NULL) {
perror("malloc:");
exit(-1);
}
daddr.nl_family = AF_NETLINK;
daddr.nl_pid = 0;
daddr.nl_groups = 0;

sd = socket(PF_NETLINK, SOCK_DGRAM, NETLINK_FSEVENT);

l_local.nl_family = AF_NETLINK;
l_local.nl_groups = 0;
l_local.nl_pid = getpid();

if (bind(sd, (struct sockaddr *)&l_local,
sizeof(struct sockaddr_nl)) == -1) {
perror("bind");
close(sd);
return -1;
}
//set_fsevent_filter(FSEVENT_MOUNT|FSEVENT_UMOUNT,
set_fsevent_filter(0xffffffff,
FSEVENT_FILTER_LISTEN, FSEVENT_FILTER_ALL, 0);
printf("filesystem event: turn on filesystem event listening.\n");
while (1) {
ret = get_fsevent();
if (ret == 0) {
printf("Exit.\n");
exit(0);
}
if (ret == -1) {
perror("recvmmsg:");
exit(1);
}
fsevent = (struct fsevent *)NLMSG_DATA(nlhdr);
type = fsevent->type;
if (MATCH_FSEVENT_TYPE(type, FSEVENT_ACCESS))
print_fsevent(fsevent, "access", 0);
```

[2.6.18 PATCH]: Filesystem Event Reporter V4 -- test program

```
if (MATCH_FSEVENT_TYPE(type, FSEVENT_MODIFY))
print_fsevent(fsevent, "modify", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_MODIFY_ATTRIB))
print_fsevent(fsevent, "modify_attr", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_CLOSE))
print_fsevent(fsevent, "close", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_OPEN))
print_fsevent(fsevent, "open", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_MOVE))
print_fsevent(fsevent, "move", 1);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_CREATE))
print_fsevent(fsevent, "create", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_DELETE))
print_fsevent(fsevent, "delete", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_MOUNT))
print_fsevent(fsevent, "mount", 1);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_UMOUNT))
print_fsevent(fsevent, "umount", 0);
if (MATCH_FSEVENT_TYPE(type, FSEVENT_FILTER_ALL))
print_ack("all");
if (MATCH_FSEVENT_TYPE(type, FSEVENT_FILTER_PID))
print_ack("pid");
if (MATCH_FSEVENT_TYPE(type, FSEVENT_FILTER_UID))
print_ack("uid");
if (MATCH_FSEVENT_TYPE(type, FSEVENT_FILTER_GID))
print_ack("gid");
printf("event type = %08x, pid = %d, uid = %d, gid = %d\n",
fsevent->type, fsevent->pid,
fsevent->uid, fsevent->gid);
printf("counter = %d\n\n", counter++);
}
}
////////////////////////////////////
```

—  
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in  
the body of a message to majordomo@xxxxxxxxxxxxxxxxx  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
Please read the FAQ at <http://www.tux.org/lkml/>