```
**********************************************************
    5082 Fri Nov  8 15:16:42 2013
new/usr/src/cmd/krb5/kwarn/kwarndtest.c
4307 Variable is used before it is initialized in kwarndtest.c
Reviewed by: Igor Kozhukhov <ikozhukhov@gmail.com>
**********************************************************
/*
 * Copyright 1995-2002 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2013 Nexenta Systems. All rights reserved.
#endif /* ! codereview */
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Test client for kwarnd.  This program is not shipped on the binary
 * release. This code was taken and modified from gssdtest.c
 */

#include <stdio.h>
#include <strings.h>
#include <ctype.h>
#include <stdlib.h>
#include "kwarnd.h"
#include <rpc/rpc.h>

#define LOOP_COUNTER  100

#define OCTAL_MACRO "%03.3o."
#define MALLOC(n) malloc(n)
#define CALLOC(n, s) calloc((n), (s))
#define FREE(x, n) free(x)

static void instructs(void);
static void usage(void);
static int parse_input_line(char *, int *, char ***);
extern uid_t getuid(void);

static void _kwarnd_add_warning(int, char **);
static void _kwarnd_del_warning(int, char **);

static int do_kwarndtest(char *buf);

extern OM_UINT32 kwarn_add_warning();
extern OM_UINT32 kwarn_del_warning();

static int read_line(char *buf, int size)
{
        int len;

        /* read the next line. If cntl-d, return with zero char count */
        printf(gettext("\n> "));

        if (fgets(buf, size, stdin) == NULL)
                return (0);

        len = strlen(buf);
        buf[--len] = '\0';
        return (len);
}
______unchanged_portion_omitted_

static int
do_kwarndtest(char *buf)
```

```
{
        int argc;
        char **argv, **argv_array;

        char *cmd;

        argv = 0;

        if (parse_input_line(buf, &argc, &argv) == 0) {
                printf(gettext("\n"));
                return (1);
        }

        if (argc == 0) {
                usage();
                FREE(argv, (argc+1)*sizeof (char *));
                /*LINTED*/
                FREE(argv_array, (argc+1)*sizeof (char *));
                return (0);
        }

        /*
         * remember argv_array address, which is memory calloc'd by
         * parse_input_line, so it can be free'd at the end of the loop.
         */

        argv_array = argv;

        cmd = argv[0];

        argc--;
        argv++;

        if (strcmp(cmd, "kwarn_add_warning") == 0 ||
            strcmp(cmd, "add") == 0) {
                _kwarnd_add_warning(argc, argv);
        } else if (strcmp(cmd, "kwarn_del_warning") == 0 ||
            strcmp(cmd, "delete") == 0) {
                _kwarnd_del_warning(argc, argv);
        } else if (strcmp(cmd, "exit") == 0) {
                printf(gettext("\n"));
                FREE(argv_array, (argc+2) * sizeof (char *));
                return (1);
        } else
                usage();

        /* free argv array */

        FREE(argv_array, (argc+2) * sizeof (char *));
        return (0);
}
______unchanged_portion_omitted_
```