```c
// $Id: strlist.c,v 1.4 2012-11-08 18:38:10-08 - - $

// Reads in a sequence of lines and then prints them out in debug
// format.  strdup(3) copies these lines onto the heap.  Read the
// comments in the file ‘numlist.c’ first.

#include <assert.h>
#include <libgen.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//
// Declaration for linked list of nodes.
//
typedef struct node *node_ref;
struct node {
   char *string;
   node_ref link;
};

int main (int argc, char **argv) {
   char *progname = basename (argv[0]);
   node_ref head = NULL;
   char buffer[256];
   int linenr;
   for (linenr = 1; ; ++linenr) {

      // Read a line of input and check to see if it ends with
      // a newline character.  Print a message if not.

      char *gotline = fgets (buffer, sizeof buffer, stdin);
      if (gotline == NULL) break;

      char *nlpos = strchr (buffer, ’\n’);
      if (nlpos != NULL) {
         *nlpos = ’\0’;
      }else {
         fprintf (stderr, "%s: %d: unterminated line: %s\n",
                  progname, linenr, buffer);
      };

      // Allocate a node and initialize it to point a a heap copy
      // of the input line.  Note that strdup(3) contains a call
      // to malloc(3), so we need the NULL check there as well.

      node_ref new = malloc (sizeof (struct node));
      assert (new != NULL);
      new->string = strdup (buffer);
      assert (new->string != NULL);
      new->link = head;
      head = new;
   };

   // Print the results in debug mode.

   printf ("%s: head= %p\n", argv[0], (void*) head);
   while (head != NULL) {
      node_ref old = head;
      head = head->link;
      printf ("%s: %p-> node {\n"
              "    string= %p->\"%s\",\n"
              "    link= %p}\n",
              progname, (void*) old, (void*) old->string,
              old->string, (void*) old->link);
   };

   return EXIT_SUCCESS;
}

/*
//TEST// (echo "this is line 1" \
//TEST// ;echo "" \
//TEST// ;echo "the previous line has length 0." \
//TEST// ;echo "fit the buffer." \
//TEST// ;echo "Last Line." \
//TEST// ) | valgrind --leak-check=full --log-file=strlist.lisval \
//TEST// ./strlist >strlist.lisout 2>&1
//TEST// mkpspdf strlist.ps strlist.c* strlist.lis*
*/

```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting strlist.c
strlist.c: $Id: strlist.c,v 1.4 2012-11-08 18:38:10-08 - - $
gcc -g -O0 -Wall -Wextra -std=gnu99 strlist.c -o strlist -lm
strlist.c: In function 'main':
strlist.c:22: warning: unused parameter 'argc'
rm -f strlist.o
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished strlist.c
```

```
./strlist: head= 0x4c232d0
strlist: 0x4c232d0-> node {
    string= 0x4c23320->"Last Line.",
    link= 0x4c23230}
strlist: 0x4c23230-> node {
    string= 0x4c23280->"fit the buffer.",
    link= 0x4c23180}
strlist: 0x4c23180-> node {
    string= 0x4c231d0->"the previous line has length 0.",
    link= 0x4c230e0}
strlist: 0x4c230e0-> node {
    string= 0x4c23130->"",
    link= 0x4c23040}
strlist: 0x4c23040-> node {
    string= 0x4c23090->"this is line 1",
    link= (nil)}
```

```
==2766== Memcheck, a memory error detector
==2766== Copyright (C) 2002-2010, and GNU GPL'd, by Julian Seward et al.
==2766== Using Valgrind-3.6.0 and LibVEX; rerun with -h for copyright info
==2766== Command: ./strlist
==2766== Parent PID: 2763
==2766==
==2766==
==2766== HEAP SUMMARY:
==2766==     in use at exit: 155 bytes in 10 blocks
==2766==   total heap usage: 10 allocs, 0 frees, 155 bytes allocated
==2766==
==2766== 155 (16 direct, 139 indirect) bytes in 1 blocks are definitely lost in loss record 3 of 3
==2766==    at 0x4A05FDE: malloc (vg_replace_malloc.c:236)
==2766==    by 0x40083D: main (strlist.c:47)
==2766==
==2766== LEAK SUMMARY:
==2766==    definitely lost: 16 bytes in 1 blocks
==2766==    indirectly lost: 139 bytes in 9 blocks
==2766==      possibly lost: 0 bytes in 0 blocks
==2766==    still reachable: 0 bytes in 0 blocks
==2766==         suppressed: 0 bytes in 0 blocks
==2766==
==2766== For counts of detected and suppressed errors, rerun with: -v
==2766== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 6 from 6)
```