```
#include "ucode.c"

char *cp, mytty[32];
char uline[2048], buf[1024], zero;

int str(char *src, char *target)
{
  int i;
  for (i = 0; i < strlen(src) - strlen(target); i++)
  {
    if (strncmp(&src[i], target, strlen(target)) == 0)
    {
      // printf("line=%s\n", src);
      return 1;
    }
  }
  return 0;
}

int main(int argc, char *argv[])
{
  int fd, n, count, cr, i, j, newline, backspace;
  int lineRead;

  STAT st0, st1, sttty;
  int redirect;

  print2f("Now entering my grep\n\r");

  cr = '\r';
  newline = '\n';
  backspace = '\b';
  gettty(mytty);

  /*****
    printf("mytty=%s\n", mytty);
    getc();
    *******/
  fstat(0, &st0);
  fstat(1, &st1);
  stat(mytty, &sttty);

  redirect = 1;
  //checks if input is from terminal, if yes sets redirect to 0
  if (st0.st_dev == sttty.st_dev && st0.st_ino == sttty.st_ino)
    redirect = 0;

  if (argc < 2)
  { // grep from stdin
    printf("usage : grep pattern filename\n");
    exit(1);
  }
```

```c
  if (argc == 2)
  {


    // if 0 has been redirected ==> do NOT show the lines read==>getline()
    // otherwise, must show each char typed ==> call gets()
    if (redirect)
    {
      //grep for piping:
      lineRead = 1;
      //while there is a next line
      while (line)
      {
        lineRead = getline(uline);
        //printf("line=%x\n", line);

        //check to see if argv[1] is in uline,
        //if true, print the line
        if (str(uline, argv[1]))
          printf("%s", uline);
      }
    }
    else
    {
      //while the user is typing
      while (gets(uline))
      {
        //if what the user typed contains argv[1], echo
        if (str(uline, argv[1]))
          printf("%s", uline);
      }
    }
  }
  else
  {
    //standalone grep: used as " grep 'word' *file* "
    printf("open %s for read\n", argv[2]);
    fd = open(argv[2], O_RDONLY); /* open input file for READ */
    if (fd < 0)
    {
      printf("open %s failed\n", argv[2]);
      exit(2);
    }

    count = 0;
    while ((n = read(fd, buf, 1024)))
    {
      buf[n] = 0;
      uline[0] = 0;

      //printf("buf=%s", buf);
      j = 0;

      for (i = 0; i < n; i++)
      {
```

```
    if (buf[i] == '\n' || buf[i] == '\r')
      break;
    uline[j++] = buf[i];
    count++;
   }
   uline[j] = 0;
   count++;

   //printf("uline=%s\n", uline);

   //checks line for argv[1]
   if (str(uline, argv[1]))
    printf("%s", uline);

   lseek(fd, (long)count, 0);
  }
 }
}

```