The system function evaluates the string argument passed to it as if it had been typed at the command line. The standard output from the command is not captured by the program and is instead printed on the screen.

The standard output from a system command or program can be captured using a pipe. Pipes follow the same syntax as regular file functions, allowing reading, write and bidirectional connections. For example, the contents of the current directory can be read into a program using,

```
#include <stdio.h>
int main() {
  int c;
  FILE *ptr = 0; /* Create a null FILE pointer */
  ptr = popen("ls ./", "r"); /* List the files in the directory and listen */
  if(!ptr) return 1; /* If the command fails return failure. */
  while((c=fgetc(ptr)) != EOF) { /* Read each character. */
    printf("%c",(char)c); /* Print the characters. */
  }
  pclose(ptr); /* Close the pipe */
  return 0; /* Return success to the operating system. */
}
```

In this case, each file name returned is available within the program.

Any command that can be typed at the command line can be executed using system or popen. Rather than just call simple shell functions, these command can be used to plot data using gnuplot,

```
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char *argv[]) {
  int x_min = 0, x_max = 4; /* Set the range for the plot. */
  char commandStr[100], systemCmd[200];
  if(argc < 2) {
    printf("Usage %s <function>\n", argv[0]); /* One argument is needed.*/
    return 1; /* Return error. */
  }
  /* Build the command in two steps to show what is going on. */
  sprintf(commandStr, "plot [x=%d:%d] %s(x)", x_min, x_max, argv[1]);

  /* Run the command so that gnuplot stays open. */
  sprintf(systemCmd,"echo \"%s\" | gnuplot --persist",commandStr);
  system(systemCmd); /* Tell gnuplot to plot it. */
  return 0; /* Return success to the operating system. */
}
```

Before trying this example, gnuplot should be installed by typing:

```
sudo apt-get gnuplot-x11
```

Then once the program has been compiled try, ./gplot sin The --persist flag causes the gnuplot window to stay open after the program has finished. More information on the gnuplot program is available at, http://www.gnuplot.info/

Monitoring a LINUX system

There are several useful functions which are available under LINUX, but are not implemented in the same way on other operating systems. For example, the status of the memory can be retrieved using the sysinfo,

```
#include <stdio.h>
#include <sys/sysinfo.h>
int main() {
  struct sysinfo info; /* Create a sysinfo instance to hold the result. */
  sysinfo(&info); /* Get the system information */
  printf("Memory used = %d\n",info.totalram - info.freeram);
  return 0; /* Return success to the operating system. */
}
```