Subject: [PATCH 1/4]: Support multiple pipe test cases
Posted by Sukadev Bhattiprolu on Tue, 24 Jun 2008 03:32:39 GMT
View Forum Message <> Reply to Message

```
>From a99deb9bcdd611c52589fa733dd90057f1f134bf Mon Sep 17 00:00:00 2001
From: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
Date: Sun, 22 Jun 2008 21:05:48 -0700
Subject: [PATCH] Support multiple pipe test cases

Modify pipe.c to support multiple test cases and to select a test case
using the -t option.

Implement two tests:
 - empty pipe
 - single write to pipe followed by continous read
---
 tests/pipe.c |   93 +++++++++++++++++++++++++++++++++++++++++++++++-----------
 1 files changed, 75 insertions(+), 18 deletions(-)

diff --git a/tests/pipe.c b/tests/pipe.c
index 0812cb3..76be6cc 100644
--- a/tests/pipe.c
+++ b/tests/pipe.c
@@ -1,52 +1,109 @@
 #include <stdio.h>
 #include <sys/types.h>
+#include <sys/fcntl.h>
 #include <unistd.h>
 #include <stdlib.h>
 #include <string.h>
 #include <errno.h>
-#include <sys/fcntl.h>

-int main(int argc, char *argv[])
+char *test_descriptions[] = {
+ NULL,
+ "Test with an empty pipe",
+ "Test with single-write to and continous read from a pipe",
+ "Test continous reads/writes from pipe",
+ "Test non-consecutive pipe-fds",
+ "Test with read-fd > write-fd",
+ "Test with read-fd/write-fd swapped",
+ "Test with all-fds in use",
+ "Test with all-fds in use for pipes",
+};
+
+static int last_num = 2;
+usage(char *argv[])
```

```
+{
+ int i;
+ printf("Usage: %s -t <test-number>\n", argv[0]);
+ printf("\t where <test-number is in range 1..%d\n", last_num);
+ for (i = 0; i < last_num; i++)
+  printf("\t test-%d: %s\n", i, test_descriptions[i]);
+ exit(1);
+}
+
+int test1()
 {
  int i = 0;
- int rc;
  int fds[2];
+
+ if (pipe(fds) < 0) {
+  perror("pipe()");
+  exit(1);
+ }
+
+ printf("Running as %d\n", getpid());
+ while (i<100) {
+  sleep(1);
+  printf("i is %d (pid %d)\n", i, getpid());
+  i++;
+ }
+}
+
+int test2()
+{
+ int i = 0;
  int c;
- int empty;
+ int rc;
+ int fds[2];
  char *buf = "abcdefghijklmnopqrstuvwxyz";

- /*
-  * -e: test with an empty pipe
-  */
- empty = 0;
- if (argc > 1 && strcmp(argv[1], "-e") == 0)
-  empty = 1;
-
  if (pipe(fds) < 0) {
   perror("pipe()");
   exit(1);
  }
```

```
- if (!empty)
-  write(fds[1], buf, strlen(buf));
+ write(fds[1], buf, strlen(buf));

  if (fcntl(fds[0], F_SETFL, O_NONBLOCK) < 0) {
   perror("fcntl()");
   exit(1);
  }
+
  printf("Running as %d\n", getpid());
  while (i<100) {
   sleep(1);
   if (i%5 == 0) {
    c = errno = 0;
    rc = read(fds[0], &c, 1);
-   if (rc != 1) {
-    perror("read() failed");
-   }
-   printf("i is %d (pid %d), c is %c\n", i, getpid(), c);
-
+   if (rc != 1)
+    perror("read() pipe failed\n");
+   printf("i is %d (pid %d), next byte is %d\n", i,
+     getpid(), c);
   }
   i++;
  }
 }

+int
+main(int argc, char *argv[])
+{
+ int c;
+ int tc_num;
+
+ while((c = getopt(argc, argv, "t:")) != EOF) {
+  switch(c) {
+  case 't':
+   tc_num = atoi(optarg);
+   break;
+  default:
+   printf("Unknown option %c\n", c);
+   usage(argv);
+  }
+ }
+
+ switch(tc_num) {
```

```
+ case 1: test1(); break;
+ case 2: test2(); break;
+ default:
+  printf("Unsupported test case %d\n", tc_num);
+  usage(argv);
+ }
+}
--
1.5.2.5
```