ใบงาน 8b

วัตถุประสงค์ เพื่อศึกษาวิธีการรับ/ส่งข้อมูล ผ่าน pipe ระหว่างโพรเซสบนระบบ unix, ศึกษา file descriptor เบื้องต้น

```
1  #include <stdlib.h> //for atoi()
2  #include <unistd.h> //for pipe
3  #include <wait.h>   //for wait()
4  #include <string.h> //for strcpy()
5  #define SIZE 10     //pipe buffer size
6  #include <stdio.h>
7
8  int main() {
9    int pfd[2];
10   //storing (pipe) file descriptor
11   //returned from pipe()
12   int nread;
13   int pid;  //pid_t is actually an int
14   char buf[SIZE];
15   char inbuf[SIZE*2];
16   pipe(/* q1.1 */);
17   //if (pipe(pfd) == -1)
18    //{perror("pipe failed\n"); exit(-1);}
19   printf("write pipe id = %d " , pfd[1]);
20   printf(" read file id = %d\n", pfd[0]);
21
22   pid = fork();
23   if (pid == 0) { //child
24      close( pfd[1] );  //tidy unused end
25      //read until end of stream
26      while ((nread = read(pfd[0], /* q1.2 */, SIZE)) != 0)
27         if (nread > 11)
28             printf("avoid overflow no conversion %s to int",buf);
29         else
30             printf("child received %s. After + 5 =
31                                        %d\n",buf,atoi(buf) + 5);
32      close(pfd[0]);   //properly close unused resource
33   } else {
34       close(/* q1.5 */ );         //tidy unused end
35       //strcpy(inbuf,"4321");     //equiv "4321\0"
36       sprintf(inbuf, "%ld", 123456789012);
37       // newer function - less side effect
38       write( /* q1.3 */ );   // +1 for padded \0
39       close( /* q1.4 */ );
40       /* q1.5 */;            //wait for child to complete
     }
     return 0;
}
```

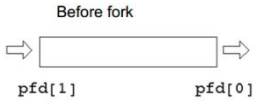
1. pipe() นัยยะหนึ่งเป็นการให้ระบบปฏิบัติการจัดการหน่วยความจำให้โพรเซสส่งข้อมูลหาอีกโพรเซสหนึ่ง ด้วยการอ่านเขียนหน่วยความจำนั้น

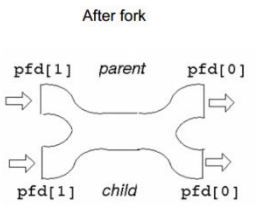
1.1 สร้าง pipe (line 16)

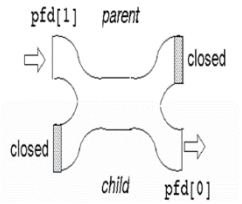
สามารถดู file descriptor ได้ (สังเกตว่าเป็นค่ามากกว่า 2) แล้ว fork เพื่อให้ทั้งสองโพรเซสเห็น pipe เดียวกัน แล้ว ปิดปลายที่ไม่ได้ใช้

1.2 การส่งคือการนำข้อมูลจากหน่วยความจำไปยัง pipe (mailbox) ทำนองเดียวกัน ผู้อ่านก็อ่านไปยังหน่วยความจำคนละที่ที่ผู้ส่งใช้ (อย่าลืมว่า fork แล้วอย่างไรต่างก็เห็นเป็นตัวแปรคนละตัว)

1.3 line 36 แสดงกรณี ข้อมูลที่ส่งยาวกว่าขนาดของ buf ที่รับข้อมูล







2. dup2()

dup2() เปลี่ยน file descriptor เป็น file descriptor ที่ต้องการ

unix reserved file descriptor ไว้ 3 เลข ได้แก่ stdin มี fd เป็น 0 stdout มี fd เป็น 1 (stderr มี fd เป็น 2)

กล่าวคือ มองคีย์บอร์ดเป็น file และ หน้าจอเป็น file ดังนั้นหากนักพัฒนาโปรแกรม map file descriptor เข้ากับ 1 การ printf() แทนที่จะออกหน้าจอ จะไปที่ไฟล์นั้นแทน

สามารถนำไปใช้ในการ pipe ได้ เช่น output จากโปรแกรมหนึ่งไปเป็น input ของอีกโปรแกรมหนึ่ง

สามารถทำ pipe ที่ shell ด้วย operator vertical bar | เช่น ls –l | wc คือ output ของ ls จะไปเป็น input ของ wc (word count) (ผลลัพธ์ตัวอย่าง) 3 บรรทัด 20 คำ 95 ตัวอักษร

อีก operator ที่เป็นประโยชน์ คือ redirect > โดย output ของการ redirect มักจะเป็น content ของ file descriptor (เช่น unix จะแทน 2 เป็น stderr ทันที)

```
ban@DESKTOP-EOON606:~$ ls -l
total 0
d--------- 1 ban ban 4096 Mar 11 20:41 OS
drwxrwxrwx 1 ban ban 4096 Mar 11 20:53 os622
ban@DESKTOP-EOON606:~$ ls -l | wc
   3   20   95
ban@DESKTOP-EOON606:~$ ls -l | wc > aaa.txt
ban@DESKTOP-EOON606:~$ cat aaa.txt
   4   29  142
ban@DESKTOP-EOON606:~$
```

ให้นักศึกษา สร้าง Lab8_2.sh โดยให้เรียก q2 ซึ่งสร้างโดย Lab8b_q2.c และใช้คำสั่ง cat aaa.txt เพื่อดูผลการทำงานของ .c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
  FILE *file_desc = fopen("aaa.txt","w");
  //"w" write mode
  int fd = fileno(file_desc);
  //obtain file descriptor
  printf("current file descriptor id is %d\n",fd);

  // after mapping to stdout, texts to be displayed
  // will be redirected to aaa.txt instead
  /* ans2.1 */

  /* to aaa.txt instead of screen */
  printf("please read this line in aaa.txt\n");
  close(fd);
  return 0;
}
```

```
prompt> gcc -o q2 lab8b_q2.c
prompt> echo "./q2" > Lab8b_2.sh
prompt> chmod +x Lab8b_2.sh
prompt> echo cat aaa.txt >> Lab8b_2.sh
prompt> ./Lab8b_2.sh
```

คำสั่ง

A. ตอบ q1.3 – q1.8

B. capture ผลลัพธ์ Lab8b_q1.c

C. ตอบ ans2.1(Lab8b_q1.c)

D. capture ผลลัพธ์ Lab8b_2.sh