

Operating Systems Lab Report

Muhammad Shafeen

Student ID: 22P-9278

October 18, 2024

Lab 8: IPC with Pipes

0.1 6.1 : Pipe on the Shell

Using pstree to see the directories tree.

```

shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ pstree
systemd--ModemManager--2*[{ModemManager}]
--NetworkManager--2*[{NetworkManager}]
--accounts-daemon--2*[{accounts-daemon}]
--acpid
--avahi-daemon--avahi-daemon
--bluetoothd
--colord--2*[{colord}]
--cron
--cups-browsed--2*[{cups-browsed}]
--cupsd
--dbus-daemon
--fwupd--4*[{fwupd}]
--gdm3--gdm-session-work--Xorg--{Xorg}
--gnome-session-b--2*[{gnome-session-b}]
--2*[{gdm-x-session}]
--2*[{gdm3}]
--gnome-keyring-d--3*[{gnome-keyring-d}]
--irqbalance--{irqbalance}
--2*[{kerneloops}]
--mount.ntfs
--networkd-dispatcher
--nm-dnsmasq--3*[{nm-dnsmasq}]
  
```

Figure 1: Executing the command 'pstree' to see the treelike structure

0.2 6.1 : 'pstree — less'

we now use ' — ' pipe take the output of one process as input of another process

```

shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ pstree | less
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ pstree | less
  
```

Figure 2: Executing the command 'pstree — less' to see scrollable tree

0.3 6.1 : 'pstree — grep bash'

should print only those lines of text from the pstree output in which the keyword bash appears

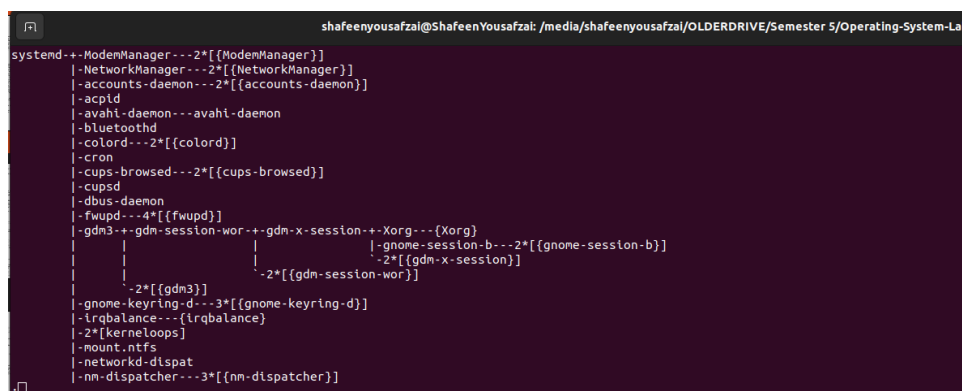


Figure 3: Only showing output with the word bash

0.4 6.1 : 'pstree — grep bash'

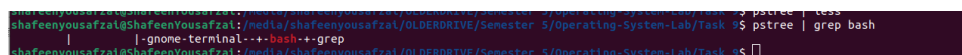


Figure 4: Output of the above command

0.5 6.1 : Code

```
#include <unistd.h>
int main()
{
    int pfd[2];
    pipe(pfd);
}
```

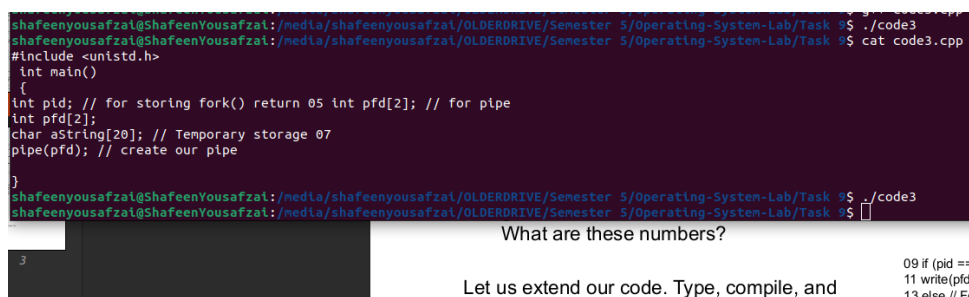


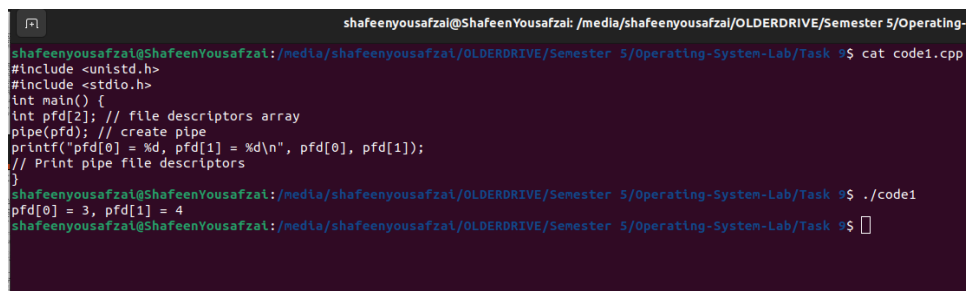
Figure 5: Only showing output with the word bash

0.6 6.1 : Task 1

```
#include <unistd.h>
```

```
#include <stdio.h>
int main() {
int pfd[2]; // file descriptors array
pipe(pfd); // create pipe
printf("pfd[0] = %d, pfd[1] = %d\n", pfd[0], pfd[1]);
// Print pipe file descriptors
}
```

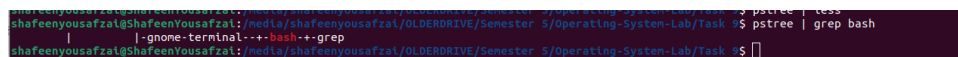
two file descriptor values, '3' and '4', represent the pipe's read and write ends respectively.



```
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-S
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task %$ cat code1.cpp
#include <unistd.h>
#include <stdio.h>
int main() {
int pfd[2]; // file descriptors array
pipe(pfd); // create pipe
printf("pfd[0] = %d, pfd[1] = %d\n", pfd[0], pfd[1]);
// Print pipe file descriptors
}
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task %$ ./code1
pfd[0] = 3, pfd[1] = 4
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task %$
```

Figure 6: Out and code of the task , printing the values

0.7 6.1 : 'pstree — grep bash'



```
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task %$ pstree | less
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task %$ pstree | grep bash
|
|_gnome-terminal--++bash--++grep
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task %$
```

Figure 7: Output of the above command

0.8 6.1 : 'Running the code'

```
#include <unistd.h>
#include <stdio.h>
int main() {
int pid;
// Process ID for fork
int pfd[2];
// Pipe file descriptors
char aString[20]; // Buffer for the parent to store data from the pipe
pipe(pfd); // Create a pipe
pid = fork(); // Fork a child process
if (pid == 0) {
// Child process\
write(pfd[1], "Hello", 5); // Write "Hello" to the pipe
} else {
// Parent process
read(pfd[0], aString, 5); // Read from the pipe into aString
}
}
```

```

shafeenyousafzal@shafeenyousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ cat code4.cpp
#include <stdio.h>
int main() {
    int pid;
    // Process ID for fork
    int pfd[2];
    // Pipe file descriptors
    char aString[20]; // Buffer for the parent to store data from the pipe
    pipe(pfd); // Create a pipe
    pid = fork(); // Fork a child process
    if (pid == 0) {
        // Child process
        write(pfd[1], "Hello", 5); // Write "Hello" to the pipe
    } else {
        // Parent process
        read(pfd[0], aString, 5); // Read from the pipe into aString
    }
}
shafeenyousafzal@shafeenyousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ ./code4

```

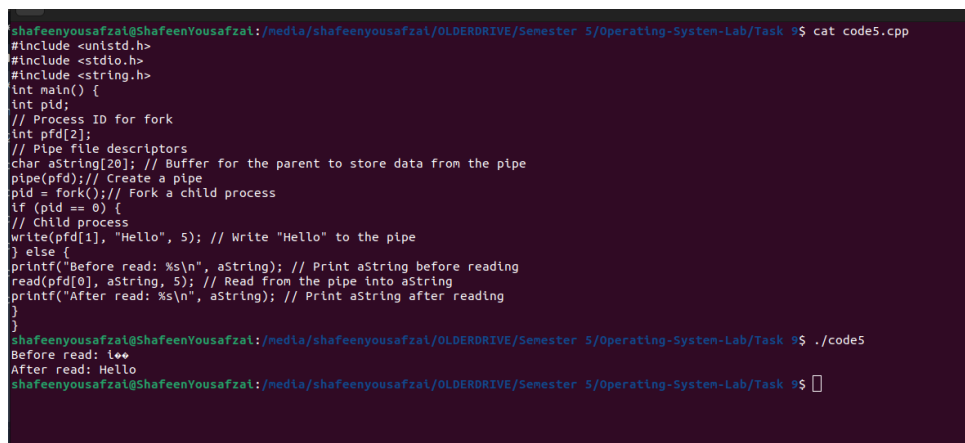
Figure 8: Running the code

0.9 6.1 : 'Task 1'

```

#include <unistd.h>
#include <stdio.h>
#include <string.h>
int main() {
    int pid;
    // Process ID for fork
    int pfd[2];
    // Pipe file descriptors
    char aString[20]; // Buffer for the parent to store data from the pipe
    pipe(pfd); // Create a pipe
    pid = fork(); // Fork a child process
    if (pid == 0) {
        // Child process
        write(pfd[1], "Hello", 5); // Write "Hello" to the pipe
    } else {
        printf("Before read: %s\n", aString); // Print aString before reading
        read(pfd[0], aString, 5); // Read from the pipe into aString
        printf("After read: %s\n", aString); // Print aString after reading
    }
}

```



```

shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ cat code5.cpp
#include <unistd.h>
#include <stdio.h>
#include <string.h>
int main() {
    int pid;
    // Process ID for fork
    int pfd[2];
    // Pipe file descriptors
    char aString[20]; // Buffer for the parent to store data from the pipe
    pipe(pfd); // Create a pipe
    pid = fork(); // Fork a child process
    if (pid == 0) {
        // Child process
        write(pfd[1], "Hello", 5); // Write "Hello" to the pipe
    } else {
        printf("Before read: %s\n", aString); // Print aString before reading
        read(pfd[0], aString, 5); // Read from the pipe into aString
        printf("After read: %s\n", aString); // Print aString after reading
    }
}
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ ./code5
Before read: 
After read: Hello
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ 

```

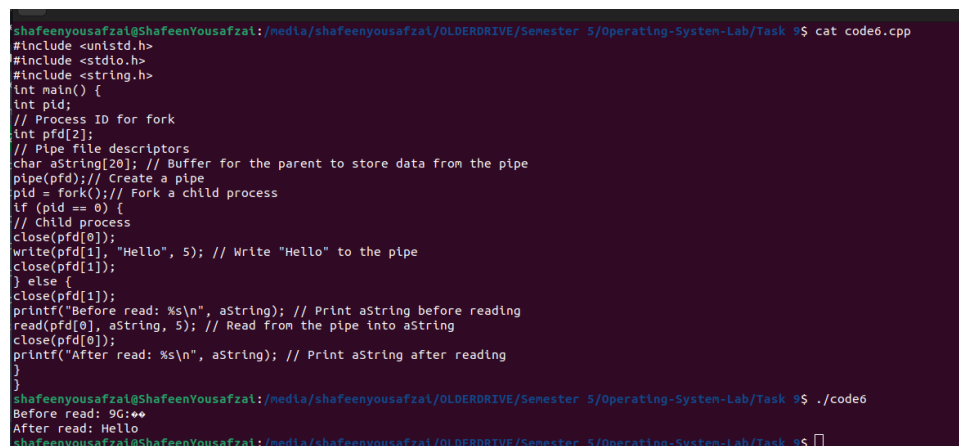
Figure 9: Running the code

0.10 6.1 : 'Task 1 Modified'

```

#include <unistd.h>
#include <stdio.h>
#include <string.h>
int main() {
    int pid;
    // Process ID for fork
    int pfd[2];
    // Pipe file descriptors
    char aString[20]; // Buffer for the parent to store data from the pipe
    pipe(pfd); // Create a pipe
    pid = fork(); // Fork a child process
    if (pid == 0) {
        // Child process
        close(pfd[0]);
        write(pfd[1], "Hello", 5); // Write "Hello" to the pipe
        close(pfd[1]);
    } else {
        close(pfd[1]);
        printf("Before read: %s\n", aString); // Print aString before reading
        read(pfd[0], aString, 5); // Read from the pipe into aString
        close(pfd[0]);
        printf("After read: %s\n", aString); // Print aString after reading
    }
}

```



```

shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ cat code6.cpp
#include <unistd.h>
#include <stdio.h>
#include <string.h>
int main() {
    int pfd;
    // Process ID for fork
    int pfd[2];
    // Pipe file descriptors
    char aString[20]; // Buffer For the parent to store data from the pipe
    pipe(pfd); // Create a pipe
    pid = fork(); // Fork a child process
    if (pid == 0) {
        // Child process
        close(pfd[0]);
        write(pfd[1], "Hello", 5); // Write "Hello" to the pipe
        close(pfd[1]);
    } else {
        // Parent process
        close(pfd[1]);
        printf("Before read: %s\n", aString); // Print aString before reading
        read(pfd[0], aString, 5); // Read from the pipe into aString
        close(pfd[0]);
        printf("After read: %s\n", aString); // Print aString after reading
    }
}
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ ./code6
Before read: 9G:♦♦
After read: Hello
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$

```

Figure 10: Running the modified code

0.11 6.1 : 'Example'

```

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main() {
    int pfd[2]; // Pipe file descriptors
    pipe(pfd);
    // Create a pipe
    if (fork() == 0) {
        // Child process
        close(pfd[1]);
        // Close the write end
        dup2(pfd[0], 0); // Redirect stdin to the read end of the pipe
        close(pfd[0]);
        // Close the read end after duplication
        execlp("wc", "wc", (char *)0); // Replace process with 'wc'
    } else {
        // Parent process
        close(pfd[0]);
        // Close the read end
        dup2(pfd[1], 1); // Redirect stdout to the write end of the pipe
        close(pfd[1]);
        // Close the write end after duplication
        execlp("ls", "ls", (char *)0); // Replace process with 'ls'
    }
    return 0;
}

```

The code says how many new lines , characters and size of the file will be displayed as output

```

@00+00+ 47W6+shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERORIVE/Semester 5/operating-System-Lab/Task 9$ cat code7.cpp
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main() {
    int pfd[2]; // Pipe file descriptors
    pipe(pfd);
    // Create a pipe
    if (fork() == 0) {
        // Child process
        close(pfd[1]);
        // Close the write end
        dup2(pfd[0], 0); // Redirect stdin to the read end of the pipe
        close(pfd[0]);
        // Close the read end after duplication
        execlp("wc", "wc", (char *)0); // Replace process with 'wc'
    } else {
        // Parent process
        close(pfd[0]);
        // Close the read end
        dup2(pfd[1], 1); // Redirect stdout to the write end of the pipe
        close(pfd[1]);
        // Close the write end after duplication
        execlp("ls", "ls", (char *)0); // Replace process with 'ls'
    }
    return 0;
}
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERORIVE/Semester 5/operating-System-Lab/Task 9$ ./code7
15      15      136
shafeenyousafzal@ShafeenYousafzal:/media/shafeenyousafzal/OLDERORIVE/Semester 5/operating-System-Lab/Task 9$

```

Figure 11: Running the example code that outputs

0.12 6.1 : 'Task 3'

```

#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include<string.h>
using namespace std;
int main()
{
    int pid;
    int pfd[2];
    char aString[20];

    pipe(pfd);

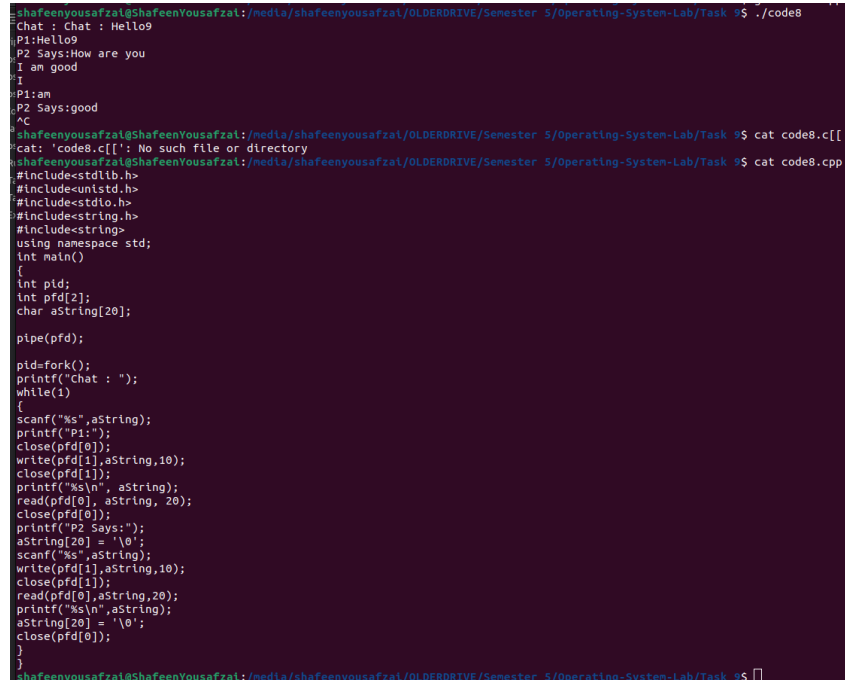
    pid=fork();
    printf("Chat : ");
    while(1)
    {
        scanf("%s",aString);
        printf("P1:");
        close(pfd[0]);
        write(pfd[1],aString,10);
        close(pfd[1]);
        printf("%s\n", aString);
        read(pfd[0], aString, 20);
        close(pfd[0]);
        printf("P2 Says:");
        aString[20] = '\0';
        scanf("%s",aString);
        write(pfd[1],aString,10);
        close(pfd[1]);
        read(pfd[0],aString,20);
        printf("%s\n",aString);
    }
}

```

```

aString[20] = '\0';
close(pfd[0]);
}
}

```



```

shafeenyousafzal@ShaFeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ ./code8
Chat : Chat : Hello9
P1:Hello9
P2 Says:How are you
I am good
P1:am
P2 Says:good
c
cat: 'code8.c[': No such file or directory
shafeenyousafzal@ShaFeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$ cat code8.cpp
#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include<string.h>
using namespace std;
int main()
{
    int pid;
    int pfd[2];
    char aString[20];

    pipe(pfd);

    pid=fork();
    printf("Chat : ");
    while(1)
    {
        scanf("%s",aString);
        printf("P1:");
        close(pfd[0]);
        write(pfd[1],aString,10);
        close(pfd[1]);
        printf("%s\n", aString);
        read(pfd[0], aString, 20);
        close(pfd[0]);
        printf("P2 Says:");
        aString[20] = '\0';
        scanf("%s",aString);
        write(pfd[1],aString,10);
        close(pfd[1]);
        read(pfd[0],aString,20);
        printf("%s\n",aString);
        aString[20] = '\0';
        close(pfd[0]);
    }
}
shafeenyousafzal@ShaFeenYousafzal:/media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 9$

```

Figure 12: Running the Task code that