HARISH K M
COE18B066

Operating System
Assignment-5
Pipes

1.Parent sets up a string which is read by child, reversed there and read back the parent

Approach:

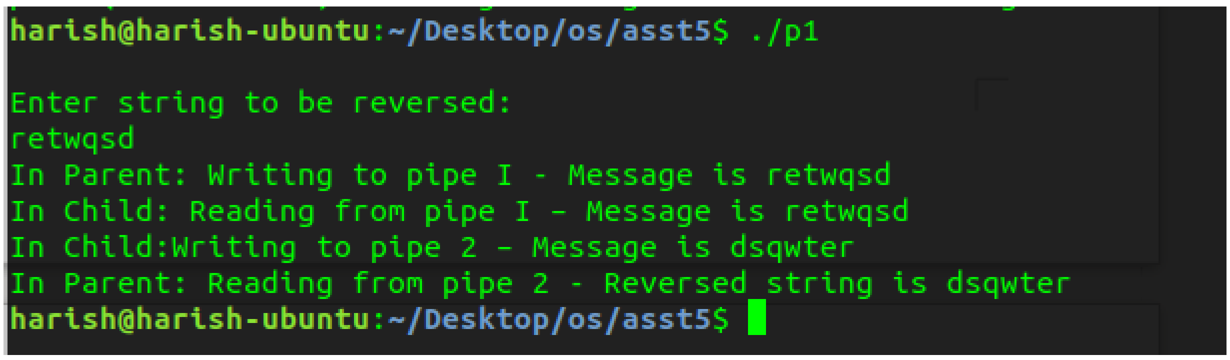
Parent receives string from user, sets the string in the pipe1.Child reads the string from pipe1,reverses it using the user defined strrev function, and writes the string in pipe2. Parent reads the reversed string and prints in on the screen.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
char *strrev(char *str)
{
if (!str || ! *str)
return str;

int i = strlen(str) - 1, j = 0;

char ch;
while (i > j)
{
ch = str[i];
str[i] = str[j];
str[j] = ch;
i--;
j++;
}
return str;
}
```

```
int main()
{
int pipefds1[2], pipefds2[2];
int returnstatus1, returnstatus2;
int pid;
char parentwrite[20];// = "string";
char readmessage[20];
returnstatus1 = pipe(pipefds1);
if (returnstatus1 == -1)
{
printf("Unable to create pipe I \n");
return 1;
}
returnstatus2 = pipe(pipefds2);
if (returnstatus2 == -1)
{
printf("Unable to create pipe 2 \n");
return 1;
}
pid = fork();
if (pid > 0) // Parent process
{
close(pipefds1[0]); // Close the unwanted pipel read side
close(pipefds2[1]); // Close the unwanted pipe2 write side
printf("\nEnter string to be reversed:\n");
gets(parentwrite);
printf("In Parent: Writing to pipe I - Message is %s\n",parentwrite);
write(pipefds1[1], parentwrite,sizeof(parentwrite)+1);
read(pipefds2[0], readmessage, sizeof(readmessage));
printf("In Parent: Reading from pipe 2 - Reversed string is %s\n",
readmessage);
}
else
{
close(pipefds1[1]); // Close the unwanted pipel write side
close(pipefds2[0]); // Close the unwanted pipe2 read side
read(pipefds1[0], readmessage, sizeof(readmessage));
printf("In Child: Reading from pipe I – Message is %s\n", readmessage);
char *reverse=strrev(readmessage);
printf("In Child:Writing to pipe 2 – Message is %s\n",reverse);
write(pipefds2[1], reverse,sizeof(reverse)+1);
return 0;
}
}
```

Output:



```
harish@harish-ubuntu:~/Desktop/os/asst5$ ./p1

Enter string to be reversed:
retwqsd
In Parent: Writing to pipe I - Message is retwqsd
In Child: Reading from pipe I – Message is retwqsd
In Child:Writing to pipe 2 – Message is dsqwter
In Parent: Reading from pipe 2 - Reversed string is dsqwter
harish@harish-ubuntu:~/Desktop/os/asst5$
```

2.Parent sets up string 1 and child sets up string 2. String 2 concatenated to string 1 at parent end and then read back at the child end.

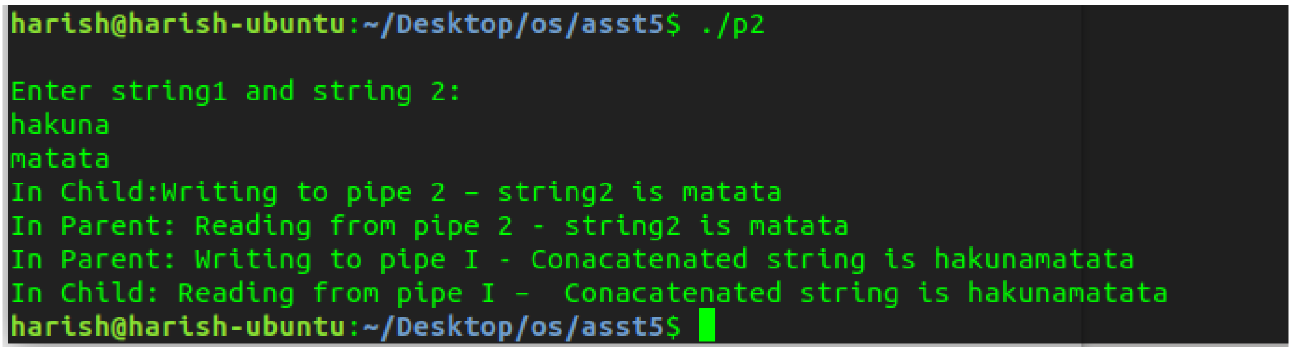
Approach:

Child receives string2 from user, sets the string in the pipe2.Parent receives string1 from user,reads the string2 from pipe2,concatenates string1 and string2 using strcat function and writes the resultant string in pipe1. Child reads the concatenated string and prints in on the screen.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
int main()
{
int pipefds1[2], pipefds2[2];
int returnstatus1, returnstatus2;
int pid;
char parentwrite[20];// = "string";
char readmessage[20];
returnstatus1 = pipe(pipefds1);
if (returnstatus1 == -1)
{
printf("Unable to create pipe I \n");
return 1;
}
returnstatus2 = pipe(pipefds2);
if (returnstatus2 == -1)
{
printf("Unable to create pipe 2 \n");
return 1;
}
pid = fork();
if (pid > 0) // Parent process
{
close(pipefds1[0]); // Close the unwanted pipel read side
close(pipefds2[1]); // Close the unwanted pipe2 write side
printf("\nEnter string1 and string 2:\n");
char s1[20],s2[20];
gets(s1);
read(pipefds2[0], s2, sizeof(s2));
strcat(s1,s2);
printf("In Parent: Reading from pipe 2 - string2 is %s\n", s2);
printf("In Parent: Writing to pipe I - Conacatenated string is %s\n",s1);
write(pipefds1[1], s1,sizeof(s1)+1);
}
else
{
close(pipefds1[1]); // Close the unwanted pipel write side
close(pipefds2[0]); // Close the unwanted pipe2 read side
//printf("\nEnter string2:\n");
char s2[20];
gets(s2);
printf("In Child:Writing to pipe 2 – string2 is %s\n",s2);
write(pipefds2[1], s2,sizeof(s2)+1);
read(pipefds1[0], readmessage, sizeof(readmessage));
printf("In Child: Reading from pipe I – Conacatenated string is %s\n",
readmessage);
return 0;
}
}
```

Output:



```
harish@harish-ubuntu:~/Desktop/os/asst5$ ./p2

Enter string1 and string 2:
hakuna
matata
In Child:Writing to pipe 2 – string2 is matata
In Parent: Reading from pipe 2 - string2 is matata
In Parent: Writing to pipe I - Conacatenated string is hakunamatata
In Child: Reading from pipe I –  Conacatenated string is hakunamatata
harish@harish-ubuntu:~/Desktop/os/asst5$
```

3.Substring generation at child end of a string setup at parent process end.

Approach:

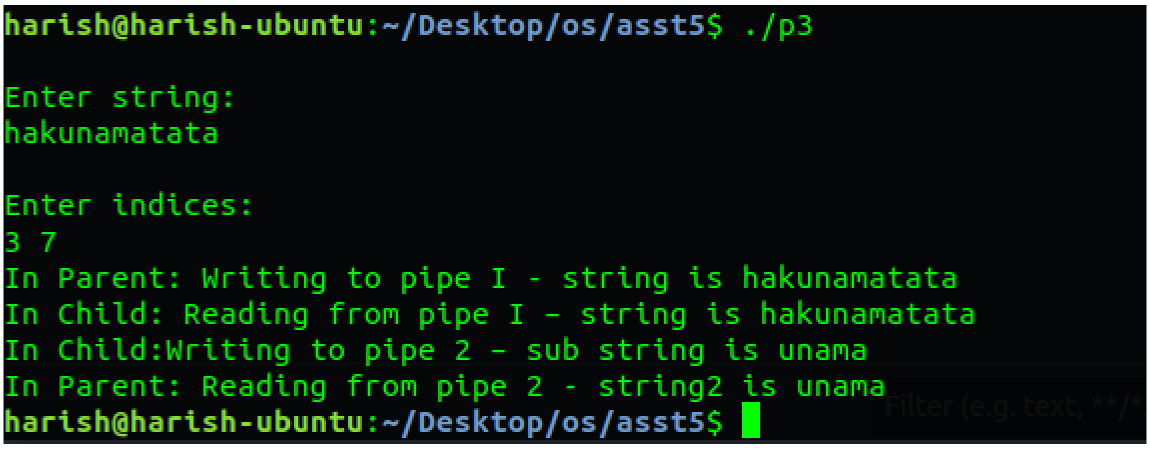
The string to be set up is received from user along with start and end of substring required.Parent sets the string in the pipe1.Child reads the string from pipe1,generates the substring required using the indices and writes the substring in pipe2. Parent reads the substring and prints in on the screen.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<stdlib.h>
int main()
{
int pipefds1[2], pipefds2[2];
int returnstatus1, returnstatus2;
int pid;
```

```
char parentwrite[20];// = "string";
char readmessage[20];

printf("\nEnter string:\n");
char s1[20],s2[20];
gets(s1);
int b,e;
printf("\nEnter indices:\n");
scanf("%d %d",&b,&e);
returnstatus1 = pipe(pipefds1);
if (returnstatus1 == -1)
{
printf("Unable to create pipe I \n");
return 1;
}
returnstatus2 = pipe(pipefds2);
if (returnstatus2 == -1)
{
printf("Unable to create pipe 2 \n");
return 1;
}
pid = fork();
if (pid > 0) // Parent process
{
close(pipefds1[0]); // Close the unwanted pipel read side
close(pipefds2[1]); // Close the unwanted pipe2 write side
printf("In Parent: Writing to pipe I - string is %s\n",s1);
write(pipefds1[1], s1,sizeof(s1)+1);
read(pipefds2[0], s2, sizeof(s2));
strcat(s1,s2);
printf("In Parent: Reading from pipe 2 - sub string is %s\n", s2);
}
else
{
close(pipefds1[1]); // Close the unwanted pipel write side
close(pipefds2[0]); // Close the unwanted pipe2 read side
//printf("\nEnter string2:\n");
//char s2[20];
read(pipefds1[0],readmessage, sizeof(readmessage));
printf("In Child: Reading from pipe I – string is %s\n", readmessage);
if(b>e)
{
printf("wrong indices\n");
exit(0);
}
else if(e>strlen(readmessage))
{
printf("wrong indices\n");
exit(0);
}
int i;
for(i=b;i<=e;++i)
s2[i-b]=readmessage[i];
s2[i-b]='\0';
printf("In Child:Writing to pipe 2 – sub string is %s\n",s2);
write(pipefds2[1], s2,sizeof(s2)+1);
return 0;
}
}
```

Output:



```
harish@harish-ubuntu:~/Desktop/os/asst5$ ./p3

Enter string:
hakunamatata

Enter indices:
3 7
In Parent: Writing to pipe I - string is hakunamatata
In Child: Reading from pipe I – string is hakunamatata
In Child:Writing to pipe 2 – sub string is unama
In Parent: Reading from pipe 2 - string2 is unama
harish@harish-ubuntu:~/Desktop/os/asst5$
```

4.String reversal and palindrome check using pipes / shared memory.

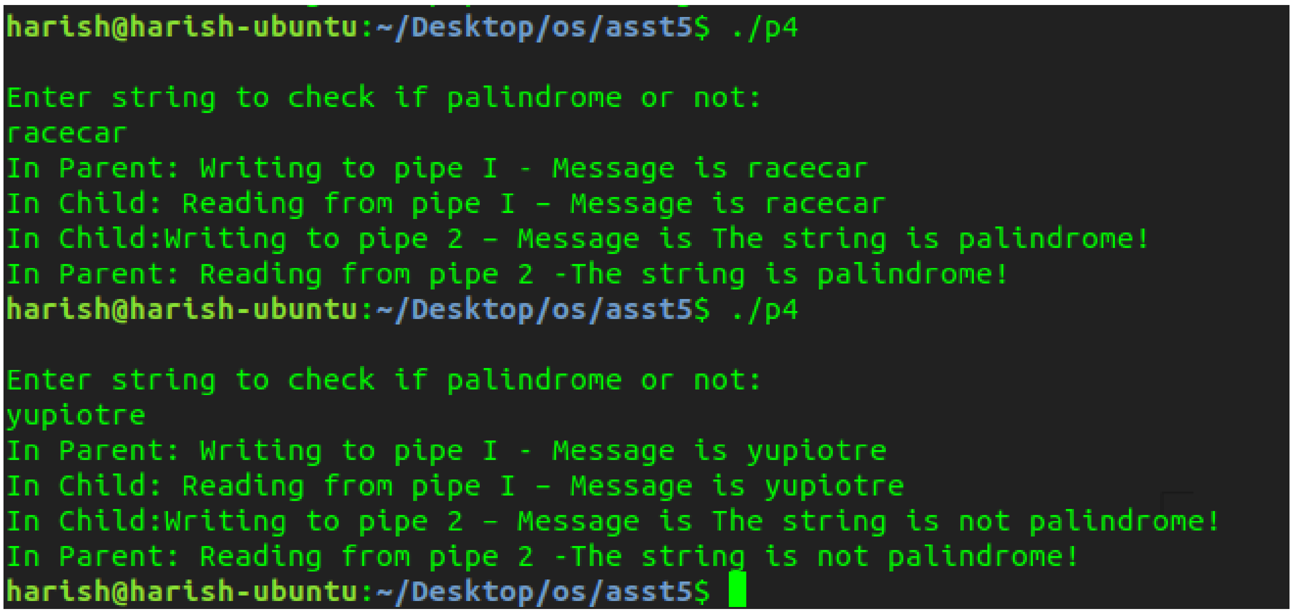
Approach:
Program receives string from user.Parent sets the string in the pipe1.Child reads the string from pipe1,reverses it using the user defined strrev function.Checks if reverse is same as original string using strcmp and writes the result whether string is palindrome or not, in pipe2. Parent reads the result and prints in on the screen.

Code:

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

void strrev(char *str)
{
if (!str || ! *str)
return;

int i = strlen(str) - 1, j = 0;

char ch;
while (i > j)
{
ch = str[i];
str[i] = str[j];
str[j] = ch;
i--;
j++;
}
return;
}

int stringcompare(char *a,char *b)
{
int flag=0;
while(*a!='\0' && *b!='\0') // while loop
{
```

```
if(*a!=*b)
{
flag=1;
}
a++;
b++;
}
if(flag==0)
return 0;
else
return 1;
}

int main()
{
int pipefds1[2], pipefds2[2];
int returnstatus1, returnstatus2;
int pid;
char parentwrite[20];
printf("\nEnter string to check if palindrome or not:\n");
gets(parentwrite);
returnstatus1 = pipe(pipefds1);
if (returnstatus1 == -1)
{
printf("Unable to create pipe I \n");
return 1;
}
returnstatus2 = pipe(pipefds2);
if (returnstatus2 == -1)
{
printf("Unable to create pipe 2 \n");
return 1;
}
pid = fork();
if (pid > 0) // Parent process
{
close(pipefds1[0]); // Close the unwanted pipel read side
close(pipefds2[1]); // Close the unwanted pipe2 write side
char readmessage[40];
printf("In Parent: Writing to pipe I - Message is %s\n",parentwrite);
write(pipefds1[1], parentwrite,sizeof(parentwrite)+1);
read(pipefds2[0], readmessage, sizeof(readmessage));
printf("In Parent: Reading from pipe 2 -%s\n", readmessage);
}
else
{
close(pipefds1[1]); // Close the unwanted pipel write side
close(pipefds2[0]); // Close the unwanted pipe2 read side
char readmessage[40];
read(pipefds1[0], readmessage, sizeof(readmessage));
printf("In Child: Reading from pipe I – Message is %s\n", readmessage);
char message[30];
strcpy(message,readmessage);
strrev(readmessage);
char result[40];
if((stringcompare(message,readmessage))!=0)
strcpy(result,"The string is not palindrome!");
else
strcpy(result,"The string is palindrome!");

printf("In Child:Writing to pipe 2 – Message is %s\n",result);
write(pipefds2[1], result,sizeof(result)+1);
//printf("%s",result);
return 0;
}
}
```

Output:



```
harish@harish-ubuntu:~/Desktop/os/asst5$ ./p4

Enter string to check if palindrome or not:
racecar
In Parent: Writing to pipe I - Message is racecar
In Child: Reading from pipe I – Message is racecar
In Child:Writing to pipe 2 – Message is The string is palindrome!
In Parent: Reading from pipe 2 -The string is palindrome!
harish@harish-ubuntu:~/Desktop/os/asst5$ ./p4

Enter string to check if palindrome or not:
yupiotre
In Parent: Writing to pipe I - Message is yupiotre
In Child: Reading from pipe I – Message is yupiotre
In Child:Writing to pipe 2 – Message is The string is not palindrome!
In Parent: Reading from pipe 2 -The string is not palindrome!
harish@harish-ubuntu:~/Desktop/os/asst5$
```