Systems Programming LAB
Lab 10

Fall 2023

Submitted by: **Hamza Mateen**

Registration No. : **21PWCSE2013**

Class Section: **C**

“As student of University of Engineering & Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: Hamza

Submitted to:

Engr. Abdullah Hamid

Jan 31, 2023

Department of Computer Systems Engineering

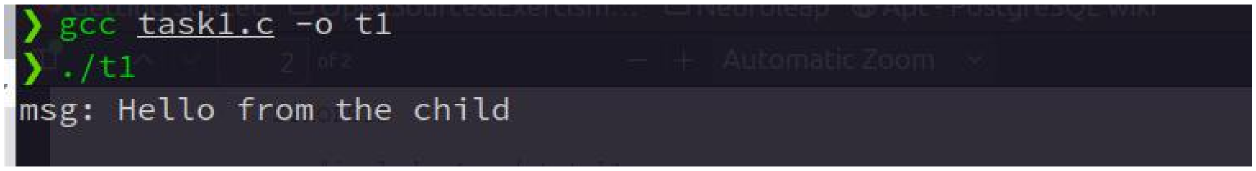
University of Engineering and Technology, Peshawar

Q NO 1: A program in which a child writes a string to a pipe and the parent reads the string.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <sys/select.h>


int main(int argc, char const *argv[])
{
// create the file desecriptors
int fd[2];
int retValue = pipe(fd);

int pid = fork();

int isChild = pid > 0;

if (isChild) {
// create a string
char* message = "Hello from the child\n";
int bw = write(fd[1], message, strlen(message));

} else {

// parent receives the message
char buffer[256];
int br = read(fd[0], buffer, 256);

printf("msg: %s", buffer);
}

return 0;
}
```

Ouput



```
❯ gcc task1.c -o t1
❯ ./t1
msg: Hello from the child
```

Q NO 2: Write a program that creates a process fan. Parent process writes to the pipe and all the child processes read the message from pipe and display it on stdout.

Code:

```c
// paretn writes a string and child displays it
// task2: fan procs, 1 parent and multiple childs and parent writes and
every
// child reads it, effectively creating a broadcast system using pipes
using
// fifos and creating a chatserver
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/select.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
// creates a process fan
// parent writes to the pipe
// all the children read the message from the pipe
int procsCount = atoi(argv[1]);
int br, bw;
char *msg = "Hello, kid!";
char buffer[100];
int fds[2], rpid;
```

```c
pipe(fds);
```

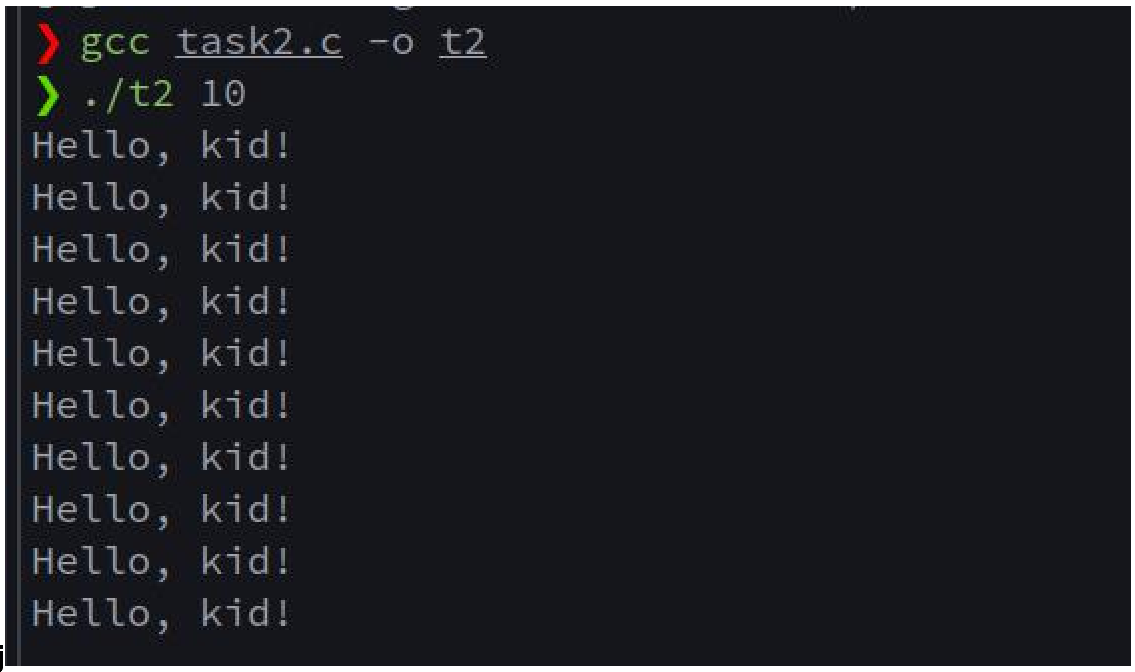
```c
// parent (main thread) writes to pipe procsCount many times
for (int i = 0; i < procsCount; i++) {
bw = write(fds[1], msg, strlen(msg));
}
// parent now reads, just to test things
for (int i = 0; i < procsCount; i++) { // child reads
```

```
rpid = fork();
```

```
if (rpid == 0) {
br = read(fds[0], buffer, strlen(msg));
buffer[strlen(msg)] = '\0';
printf("%s\n", buffer);
exit(0);
}
}
```

```
return EXIT_SUCCESS;
}
```

Ouput



```
❯ gcc task2.c -o t2
❯ ./t2 10
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
Hello, kid!
```

j

Q NO 3: Find Utility

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```c
#include <fcntl.h>
#include <string.h>
#include <sys/select.h>

#define FIFO_NAME "chat_fifo"
#define BUFFER_SIZE 256
#define TIMEOUT_SECONDS 60
```

```c
int main() {
if (access(FIFO_NAME, F_OK) == -1) {
// FIFO does not exist, create it
if (mkfifo(FIFO_NAME, 0666) == -1) {
perror("Failed to create FIFO");
exit(EXIT_FAILURE);
}
}
```

```c
int fifo_fd = open(FIFO_NAME, O_RDWR);
if (fifo_fd == -1) {
perror("Failed to open FIFO");
exit(EXIT_FAILURE);
}
```

```c
fd_set read_fds;
char buffer[BUFFER_SIZE];
```

```c
while (1) {
FD_ZERO(&read_fds);
FD_SET(STDIN_FILENO, &read_fds);
FD_SET(fifo_fd, &read_fds);
```

```c
int max_fd = (STDIN_FILENO > fifo_fd) ? STDIN_FILENO : fifo_fd;
```

```c
struct timeval timeout;
timeout.tv_sec = TIMEOUT_SECONDS;
timeout.tv_usec = 0;
```

```c
int ready = select(max_fd + 1, &read_fds, NULL, NULL, &timeout);
```

```c
if (ready == -1) {
perror("Select error");
exit(EXIT_FAILURE);
} else if (ready == 0) {
printf("No activity for %d seconds. Closing chat.\n", TIMEOUT_SECONDS);
break;
}
```

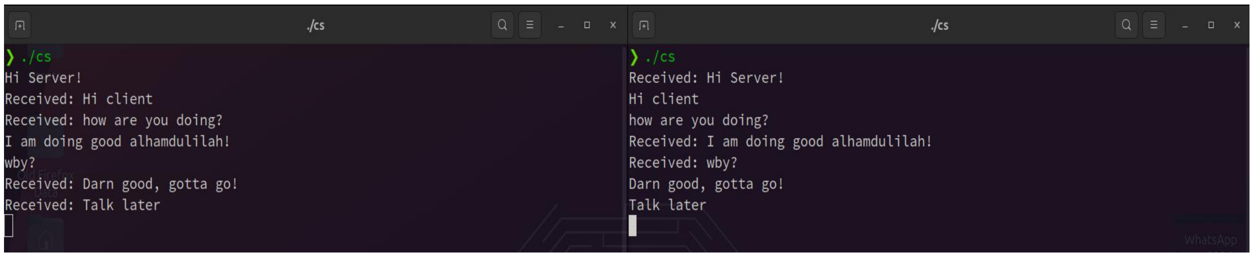
```c
if (FD_ISSET(STDIN_FILENO, &read_fds)) {
// Read from standard input and write to the FIFO
fgets(buffer, sizeof(buffer), stdin);
write(fifo_fd, buffer, strlen(buffer));
sleep(5);
}
```

```c
if (FD_ISSET(fifo_fd, &read_fds)) {
// Read from the FIFO and print to standard output
ssize_t bytesRead = read(fifo_fd, buffer, sizeof(buffer) - 1);
if (bytesRead == -1) {
perror("Read error");
exit(EXIT_FAILURE);
} else if (bytesRead == 0) {
fprintf(stderr, "Server disconnected\n");
break;
} else {
buffer[bytesRead] = '\0';
printf("Received: %s", buffer);
}
}
}
```

```c
close(fifo_fd);
unlink(FIFO_NAME);
```

```c
return 0;
}
```

Output



```
❯ ./cs
Hi Server!
Received: Hi client
Received: how are you doing?
I am doing good alhamdulilah!
wby?
Received: Darn good, gotta go!
Received: Talk later
```

```
❯ ./cs
Received: Hi Server!
Hi client
how are you doing?
Received: I am doing good alhamdulilah!
Received: wby?
Darn good, gotta go!
Talk later
```

************************* THE END **********************