

106119029 Lab 5, OS Lab

Dipesh Kafle

Question 1

- Design a program using concepts of inter-process communication ordinary pipes in which one process sends a string message to a second process, and the second process counts the number of vowels in the message and sends it back to the first process

Code for Question 1

```
#include <ctype.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

#define WRITE_END 1
#define READ_END 0

int main() {
    sem_t *sem1 = (sem_t *)mmap(0, sizeof(sem_t), PROT_READ | PROT_WRITE,
                                MAP_ANONYMOUS | MAP_SHARED, 0, 0);

    sem_t *sem2 = (sem_t *)mmap(0, sizeof(sem_t), PROT_READ | PROT_WRITE,
                                MAP_ANONYMOUS | MAP_SHARED, 0, 0);

    sem_init(sem1, 1, 0);
    sem_init(sem2, 2, 0);
    int fd[2];
    if (pipe(fd) == -1) {
        printf("Error in initiating the pipe\n");
        exit(1);
    }
```

```

pid_t pid = fork();
if (pid < 0) {
    printf("Couldnt fork\n");
    exit(2);
}
if (pid == 0) {
    sem_wait(sem1);
    char str2[1024];
    read(fd[READ_END], str2, 1024);
    close(fd[READ_END]);
    printf("Received data in child process\n");
    fflush(stdout);
    int len = strlen(str2);
    int count = 0;
    for (int i = 0; i < len; i++) {
        char c = str2[i];
        if (isalpha(c)) {
            c = tolower(c);
            if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
                count++;
            }
        }
    }
    char str3[1024];
    sprintf(str3, "The number of vowels is %d\n", count);
    write(fd[WRITE_END], str3, strlen(str3));
    printf("Sent no of vowels to parent process\n");
    fflush(stdout);
    close(fd[WRITE_END]);
    sem_post(sem2);
    exit(0);
} else {
    char str[1024] = "A quick brown fox jumps over the lazy dog";
    write(fd[WRITE_END], str, strlen(str));
    close(fd[WRITE_END]);
    printf("Sent a string from parent process: %s\n\n", str);
    fflush(stdout);
    if (sem_post(sem1) == -1) {
        printf("Failed to post\n");
        exit(0);
    }
    fflush(stdout);
    sem_wait(sem2);
    char str4[1024];

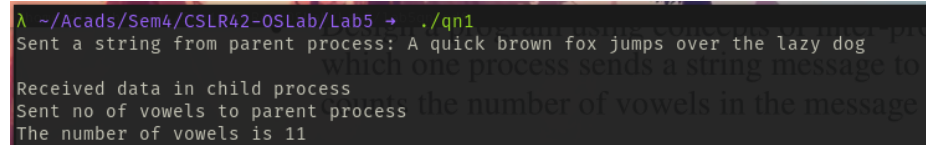
```

```

        str4[read(fd[READ_END], str4, 1023)] = 0;
        printf("%s", str4);
        fflush(stdout);
        close(fd[READ_END]);
    }
    wait(NULL);
}

```

Output for Question 1



```

λ ~/Acads/Sem4/CSLR42-OSLab/Lab5 → ./qn1
Sent a string from parent process: A quick brown fox jumps over the lazy dog
Received data in child process
Sent no of vowels to parent process
The number of vowels is 11

```

Question 2

- You will write two simple programs 'my_npipe_reader.c' and 'my_npipe_writer.c' that use a named pipe to communicate. The 'my_npipe_reader' program will set up a named pipe using 'mkfifo()', open it read only, and read strings from it until it receives the string 'exit'. The writer will open the named pipe file, read strings from the user and write them to the named pipe. When the user enters 'exit', the program will write the string to the pipe and then exit. Execution should look something like this (note that you must start the reader first)

Code for Question 2

- my_npipe_reader.c

```

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

int main() {
    printf("Creating named pipe '/tmp/my_fifo'..");
    int res = mkfifo("/tmp/my_fifo", 0666);
    if (res == -1) {
        printf("Named pipe creation failed\n");
        exit(2);
    } else {
        printf("done\n");
    }
}

```

```

}
int fd = open("/tmp/my_fifo", O_RDONLY);
if (fd == -1) {
    printf("Failed to open named pipe");
    exit(1);
}
char str[1024];
while (1) {
    printf("Waiting for input..");
    fflush(stdout);
    fflush(stdout);
    int n_bytes = read(fd, str, 1024);
    str[n_bytes] = 0;
    printf("%s\n", str);
    if (strcmp(str, "exit") == 0) {
        close(fd);
        unlink("/tmp/my_fifo");
        exit(0);
    }
}
}
}

```

- my_npipe_writer.c

```

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

int fd;

void handler(int sig) {
    int res = write(fd, "exit", strlen("exit"));
    exit(0);
}

int main() {
    signal(SIGINT, handler);
    printf("Opening named pipe /tmp/my_fifo...");
    fflush(stdout);
    // F_OK means file exists
    if (access("/tmp/my_fifo", F_OK) == 0) {
        fd = open("/tmp/my_fifo", O_WRONLY);
    }
}

```

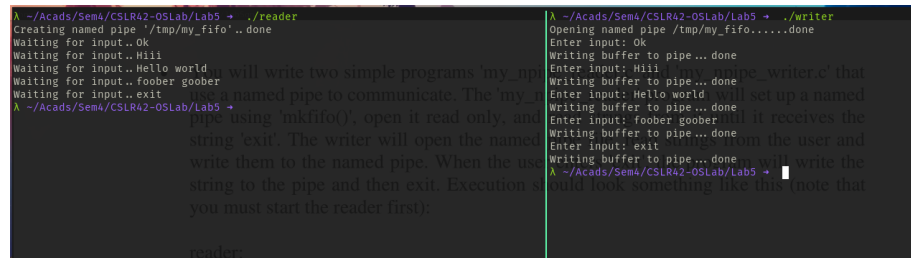
```

    if (fd == -1) {
        printf("Failed to open named pipe");
        exit(4);
    } else {
        printf("...done\n");
    }
} else {
    printf("Pipe doesnt exist yet\n");
    exit(3);
}

char buf[1024];
while (1) {
    printf("Enter input: ");
    fflush(stdout);
    buf[read(0, buf, 1023) - 1] = 0;
    /* scanf("%1023s", buf); */
    printf("Writing buffer to pipe...");
    fflush(stdout);
    int res = write(fd, buf, strlen(buf));
    if (res == -1) {
        printf("Couldnt write to the pipe\n");
        exit(1);
    } else {
        printf("done\n");
    }
    if (strcmp(buf, "exit") == 0) {
        close(fd);
        exit(0);
    }
}
}

```

Output For Question 2



The image shows two terminal windows side-by-side. The left window is running the 'reader' program, and the right window is running the 'writer' program. Both programs create a named pipe at '/tmp/my_fifo'. The 'reader' program waits for input and writes it to the pipe. The 'writer' program reads from the pipe and prints the input. The output shows the successful execution of both programs, with data being written to and read from the named pipe.

```

A ~/Acads/Sem4/CSLR42-OSLab/Lab5 → ./reader
Creating named pipe '/tmp/my_fifo'...done
Waiting for input..Ok
Waiting for input..Hiii
Waiting for input..Hello world
Waiting for input..foober goober
Waiting for input..exit
A ~/Acads/Sem4/CSLR42-OSLab/Lab5 →

A ~/Acads/Sem4/CSLR42-OSLab/Lab5 → ./writer
Opening named pipe /tmp/my_fifo.....done
Enter input: Ok
Writing buffer to pipe... done
Enter input: Hiii
Writing buffer to pipe... done
Enter input: Hello world
Writing buffer to pipe... done
Enter input: fober goober
Writing buffer to pipe... done
Enter input: exit
Writing buffer to pipe... done
A ~/Acads/Sem4/CSLR42-OSLab/Lab5 →

```