Gebze Technical University
CSE 344
SYSTEM PROGRAMMING
HOMEWORK #2

200104004066 - Ahmet Yiğit

- Introduction and Aim
    - Interprocess communication (IPC) is a fundamental concept in operating systems and parallel computing, enabling different processes to exchange data and synchronize their activities.

    - The aim of this report is to demonstrate the utilization of FIFOs for interprocess communication within a multi-process environment.

Detailed view of Homework:

- **Creating Fifo's:**

```c
int main(int argc, char *argv[])
{   // ....

    mkfifo(FIFO1, 0666);
    mkfifo(FIFO2, 0666);

    // ....
}
```

-

    - Named pipes, or FIFOs, are special files that act as a communication channel between processes.
    - They allow one-way communication between processes, where data written by one process can be read by another process.

- **Using Fifo's**

```c
int main(int argc, char *argv[]) {
    // ....

    int fd1_read = open(FIFO1, O_RDONLY |
O_NONBLOCK);
    if (fd1_read == -1) {
        unlink(FIFO1);
        error_exit("open FIFO1 in parent");
    }

    int fd1_write = open(FIFO1, O_WRONLY);
    if (fd1_write == -1) {
        unlink(FIFO1);
        error_exit("open FIFO1 in parent");
    }

    // ....
}
```

-

- I use NONBLOCK because of FIFO's are waiting to other tail with noblock it read and writes simultaneously in same process.
- Also I used with block statements because Child1 and Child2 communication must be synchronized and with FIFO's I will handle that.

```c
int main(int argc, char *argv[]) {
    // child 1

    int fd2 = open(FIFO2, O_WRONLY);
    if (fd2 == -1) error_exit("open FIFO2 in child");

    // Write result to FIFO2
    if (write(fd2, &result, sizeof(int)) == -1) {
        perror("write FIFO2 in child");
        exit(EXIT_FAILURE);
    }

    // ....

    // child 2

    int fifoMultResult = 0;
    fd2 = open(FIFO2, O_RDONLY);
    if (fd2 == -1) error_exit("open FIFO2 in child");
    // Write result to FIFO1

    if (read(fd2, &fifoMultResult, sizeof(int)) == -1) {
        perror("write FIFO1 in child");
        exit(EXIT_FAILURE);
    }

    // ....
}
```

- At the end of every FIFO end i close all of them.
- There is a size limit in FIFO's by Linux which is described in Linux manuel so same numbers not work very well on your Linux constants.
- With FIFO's communication between parent-child and child-child processes are handled.

```c
int main(int argc, char *argv[]) {
    //...
    fd2 = open(FIFO2, O_RDONLY);
    if (fd2 == -1) error_exit("open FIFO2 in child");
    // Write result to FIFO1

    if (read(fd2, &fifoMultResult, sizeof(int)) == -1) {
        perror("write FIFO1 in child");
        exit(EXIT_FAILURE);
    }

    //...
}
```

- **Forking Processes:**

  - The fork() system call is used to create child processes.

```c
int main(int argc, char *argv[])
{   // child 1

    pid_t pid = fork();

    if (pid == 0){
        // Child Process 1
        sleep(10);
    // ...
    else if(pid > 0){
      pid_t pid2 = fork();

      if(pid2 == 0){
          // Child Process 2
          sleep(10);
    // ....
}
```

  - Creating fork without an error child and parent processes are continue their execution in same time.
  - I created 2 child in that program and one parent process.

- **Signal Handling:**

  - The program sets up a signal handler for the SIGCHLD signal, which is sent to the parent process when a child process terminates.

```c
void sigchld_handler(int signo) {
    int status;
    pid_t pid;

    while ((pid = waitpid(-1, &status, WNOHANG)) > 0) {

        if (WIFEXITED(status))
            printf("Child process %d terminated - expected status: %d\n", pid, WEXITSTATUS(status));

        else
            printf("Child process %d terminated - not expected\n", pid);

        counter--;
    }

}
```

  - There is a global variable to watch counter and manage it with parent process end.

- **Cleanup**

    - When parent process ended, makefile cleaner working and clears all FIFO's and executable file also program unlinks FIFO's too.

- ## Output of Program
    - ### Making Test with N=5
        -
```
./main 5
Writing FIFO1 number: 0:4
Writing FIFO1 number: 1:2
Writing FIFO1 number: 2:5
Writing FIFO1 number: 3:4
Writing FIFO1 number: 4:7
Writting FIFO2 number: 0:5
Writting FIFO2 number: 1:3
Writting FIFO2 number: 2:6
Writting FIFO2 number: 3:5
Writting FIFO2 number: 4:8
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Reading FIFO1 number: 0:4
Reading FIFO1 number: 1:2
Reading FIFO2 number: 0:5
Reading FIFO1 number: 2:5
Reading FIFO1 number: 3:4
Reading FIFO2 number: 1:3
Reading FIFO1 number: 4:7
Result of sum from first child is: 22
Reading FIFO2 number: 2:6
Reading FIFO2 number: 3:5
Reading FIFO2 number: 4:8
Result of multiplication from second child is: 3600
WROTED FIFO2 number CHIILD1: 22
Reading FIFO2 number CHIILD1: 22
Sum of results: 3622
Child process 15171 terminated - expected status: 0
Child process 15170 terminated - expected status: 0
==========================================================================
Completed tests....
mens1s@Ahmet-MacBook-Air-3 hw2 %
```
        - This writing and reading print are not on shared code is commented!
        - When we look at results they are true.

- **Making Test With N=10**

```
Compiling...
----------------------------------------
Running the tests....
================================================================================
./main 10
Writing FIFO1 number: 0:7
Writing FIFO1 number: 1:7
Writing FIFO1 number: 2:7
Writing FIFO1 number: 3:8
Writing FIFO1 number: 4:9
Writing FIFO1 number: 5:1
Writing FIFO1 number: 6:5
Writing FIFO1 number: 7:8
Writing FIFO1 number: 8:0
Writing FIFO1 number: 9:3
Writting FIFO2 number: 0:8
Writting FIFO2 number: 1:8
Writting FIFO2 number: 2:8
Writting FIFO2 number: 3:9
Writting FIFO2 number: 4:10
Writting FIFO2 number: 5:2
Writting FIFO2 number: 6:6
Writting FIFO2 number: 7:9
Writting FIFO2 number: 8:1
Writting FIFO2 number: 9:4
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Reading FIFO1 number: 0:7
Reading FIFO1 number: 1:7
Reading FIFO1 number: 2:7
Reading FIFO2 number: 0:8
Reading FIFO1 number: 3:8
Reading FIFO1 number: 4:9
Reading FIFO1 number: 5:1
Reading FIFO2 number: 1:8
Reading FIFO1 number: 6:5
Reading FIFO2 number: 2:8
Reading FIFO2 number: 3:9
Reading FIFO1 number: 7:8
Reading FIFO2 number: 4:10
Reading FIFO2 number: 5:2
Reading FIFO2 number: 6:6
Reading FIFO2 number: 7:9
Reading FIFO2 number: 8:1
Reading FIFO1 number: 8:0
Reading FIFO2 number: 9:4
Result of multiplication from second child is: 19906560
Reading FIFO1 number: 9:3
Result of sum from first child is: 55
WROTED FIFO2 number CHIILD1: 55
Reading FIFO2 number CHIILD1: 55
Sum of results: 19906615
Child process 15434 terminated - expected status: 0
Child process 15433 terminated - expected status: 0
================================================================================
Completed tests....
mens1s@Ahmet-MacBook-Air-3 hw2 %
```

- Makefile Code
  -

```
makefile
You, 1 minute ago | 1 author (You)
all: clean compile run

compile: main.c
    @echo "----------------------------------------"
    @echo "Compiling..."
    @gcc -o main main.c
run:
    @echo "----------------------------------------"
    @echo "Running the tests...."
    @echo "==========================================================================="
    ./main 10
    @echo "==========================================================================="
    @echo "Completed tests...."

clean:
    @echo "----------------------------------------"
    @echo "Removing compiled files..."
    @rm -f *.o
    @rm -f main
    @rm -f fifo1
    @rm -f fifo2      You, 3 days ago • initial commit
```