Vežbanje 8. čas

1. Napisati program koji demonstrira dvosmernu komunikaciju između roditeljskog procesa i dete procesa korišćenjem *PIPE*-a. Roditelj sa standardnog ulaza učitava liniju po liniju komande terminala koje treba izvršiti i preko *pipe*-a ih prosleđuje detetu. Dete izvršava prosleđene komande korišćenjem *system()* funkcije (man 3 system()) i šalje *exit code* komande roditeljskom procesu - opet preko *pipe*-a. Ako je *exit code* 0, roditelj ispisuje *Success*, inače ispisuje *Failure*. Kada se unese komanda *quit* roditelj šalje signal *SIGKILL* detetu (man 2 kill), čeka na dete i završava sa radom. Pre nego što završi ispisuje poruku *Bye*.

Pokretanje programa:	./1
Standardni ulaz:	ls -l -r rm -rf / pwd quit
Standardni izlaz:	Success Failure Success Bye
Exit kod:	0

2. Program prima putanju do *FIFO* kao argument komandne linije. Iz prosleđenog *FIFO* fajla čita realne brojeve sve do kraja ulaza i na kraju ispisuje njihovu aritmetičku sredinu. U slučaju da nije pročitan nijedan broj ili da je program naisao na nesto sto nije broj (npr. neki string) završiti program sa *exit code*-om 1. Da biste testirali program otvorite dva terminala. Za detalje pogledajte tabelu sa opisom izvršavanja programa. NAPOMENA: ne morate brisati *FIFO* fajl.

Terminal 1:	mkfifo /tmp/test1 echo "1.5 -1" > /tmp/test1 rm -f /tmp/test1	mkfifo /tmp/test2 echo "bla" > /tmp/test2 rm -f /tmp/test2	mkfifo /tmp/test3 echo "0.5 bla" > /tmp/test3 rm -f /tmp/test3
Terminal 2:	./2 /tmp/test1	./2 /tmp/test2	./2 /tmp/test3
Standardni izlaz:	0.25	[redacted]	[redacted]
Exit kod:	0	1	1

3. Napisati program koji prima kao argument komandne linije putanju do C fajla. Za prosleđeni C fajl program poziva *gcc* kompajler, čeka na završetak prevođenja i ispisuje veličinu dobijenog izvršnog fajla (programa) u bajtovima. Ukoliko *gcc* ne završi sa *exit code*-om 0, program treba da završi sa *exit code*-om sa kojim je *gcc* završio izvršavanje. **Zabranjeno je korišćenje *system()* funkcije.**

4. Program kao argument komandne linije prima naziv bloka deljene memorije (eng. *shared memory*). U ovom bloku se nalazi struktura:

```
typedef struct {
     sem_t ready;
     sem_t sorted;
     double data[MAX_DATA_LEN];
     unsigned numOfElems; // number of array elements
} OsSharedDoubArray;
```

Program čeka na semafor *ready* i kada su podaci spremni sortira podatke nerastuće. Kada su podaci ispravno sortirani program postavlja semafor *sorted* kako bi obavestio neku narednu aplikaciju da su podaci sada sortirani i nakon toga završava sa izvršavanjem.

NAPOMENE: Vrednost *MAX_DATA_LEN* konstante je 1024. Memoriju inicijalizovati pozivom ***setup_4*** programa. Nakon izvšavanja vašeg programa, pozvati ***check_4*** kako biste proverili da li ste dobro uradili zadatak. Primer pozivanja:

$ ./setup_4 /4shm
$ ./4 /4shm
$ ./check_4 /4shm