Operativni sistemi Januar 2 02.02.2018.

Napraviti u **/home/ispit1** direktorijum u skladu sa indeksom i asistentom kod koga slusate kurs. Na primer, student sa indeksom 101/2016 koji slusa kurs kod Vladimira Kuzmanovica treba da napravi folder **mi16101_v**, a student sa indeksom 12/2016 koji slusa kurs kod Ognjena Kocica treba da napravi folder **mi16012_o**. Za svaki zadatak napraviti odgovarajući **.c** fajl unutar ovog foldera (1.c, 2.c ... 5.c).

Ispit se radi 3h. Svaki zadatak nosi po **20%** tj. **6 poena**. *Na izlaz za greške možete ispisivati šta god želite. Strogo se držite navedenih formata ispisa za standardni izlaz!*

Kod prevoditi sa opcijom -std=c99. Zabranjeno koriscenje *system(3)* funkcije.

1. Napisati program koji za broj sekundi od Epohe, prosledjen kao argument komandne linije, vreme u formatu **hh:mm** (odnosi se na sate i minute - pogledati pokretanja). NAPOMENA: Trenutan broj sekundi od Epohe na racunaru mozete dobiti sa komandom: *date +%s* .

Pokretanje:	./1 1515441588	./1 1515452868	./1
Std. izlaz:	20:59	00:07	----------
Exit code:	0	0	1

2. Napisati program koji sa standardnog ulaza ucitava koordinate tacaka do kraja ulaza. Svaka linija standardnog ulaza sadrzi po 2 realna broja koji predstavljaju *x* i *y* koordinate jedne tacke. Ako je ucitano *N* tacaka, pokrenuti *N* niti pri cemu *i*-ta nit racuna rastojanje *i*-te tacke od svih ostalih tacaka. Pri ovom izracunavanju, niti dodatno azuriraju globalno minimalno rastojanje izmedju bilo koje dve tacke – ovo rastojanje ispisati iz **main** funkcije.

Pokretanje:	./2

Std. ulaz:	`1.5 -2.0`
	`3.6 5.7`
	`0 -1`
	`-3.2 16.257` ← *4 tacke, dakle 4 niti ce biti pokrenute*

Std. izlaz:	1.80278 ← *kod ispisa ne brinite o broju decimalnih mesta, normalno ispisite rezultat*
Exit code:	0

3. Napisati program koji kao argument komadne linije prima putanju do fajla i broj **a**. Program pokrece dete proces koje treba da izvrsi komandu terminala ***tail*** za prosledjene argumente (npr. tail *-n 5 dir/1.txt* za argumente ‘dir/1.txt’ i ‘5’). Preusmeriti standardni izlaz iz dete procesa, procitati ga u roditeljskom procesu i ispisati na standardni izlaz. Ukoliko dete proces ne zavrsi uspesno, tj. *exit code* 0, ispisati ‘Neuspeh’ iz roditelja.

Pokretanje:	./3 neki_direktorijum/jos_jedan/fajl.bla 2	./3 ne_postoji.txt 15	./3
Fajl:	Ovo je prvi red drugi red fajla treci red fajla cetvrti i dosta je	***Nema fajla (wc ce zavrsiti sa exit code-om 1)***	------
Std. izlaz:	treci red fajla cetvrti i dosta je	Neuspeh	------
Exit code:	0	0	1

4. Napisati program koji kao argumente komandne linije prima putanje do FIFO fajlova. Program cita karaktere (bajtove) iz FIFO fajlova prateci sve fajlove istovremeno koriscenjem e*poll* interfejsa. Ispisati **naziv** FIFO fajla iz kog je procitano **najmanje** karaktera.

Pokretanje:	./4 /tmp/first_fifo /tmp/second_fifo dir/bla/third_fifo fourth_fifo	./4 /tmp/f1 /tmp/ne_postoji	./4
FIFO 1:	Ovo su neki karakteri	----------	---
FIFO 2:	Ovo je malo vise karaktera	***Nema fajla***	---
FIFO 3:	Ovo je zaista najvise karaktera	----------	---
Std. izlaz:	first_fifo	----------	---
Exit code:	0	1	1

5. Napisati program koji kao argumente komandne linije prima naziv objekta *deljene memorije*. Potrebno ucitati strukturu:

```
typedef struct {
     sem_t inDataReady;
     sem_t dataProcessed;
     char str[ARRAY_MAX];
} OsInputData;
```

i obrnuti nisku *str*. Dodatno, pre bilo kakvog obrade, potrebno je **sacekati** na semafor *inDataReady*, nakon obrade **postaviti** semafor *dataProcessed* (pretpostaviti da je ispravno inicijalizovan). NAPOMENA: Linkovati sa **-lrt**. Ne raditi *shm_unlink()*.

Pokretanje programa:	./1 /inmem	./1	./1 /nepostoji	./1 /somemem
Niska *str* pre*:*	neka_niska	------	----------	n3k4_n1sk2
Niska *str* nakon:	aksin_aken	------	----------	2ks1n_4k3n
Exit kod:	0	1	1	0

POSIX niti - dodatak

Sve funkcije za rad sa POSIX nitima vracaju pozitivnu vrednost koda greske ako je do greske doslo, a nulu inace. Zadaci koji koriste ove funkcije se moraju linkovati sa **-lpthread**. Potpisi najbitnijih funkcija slede:

```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
            void *(*start_routine) (void *), void *arg);
int pthread_join(pthread_t thread, void **retval);
int pthread_mutex_init(pthread_mutex_t *mutex,const
pthread_mutexattr_t *attr);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_cond_init(pthread_cond_t * cond, const pthread_condattr_t *attr);
int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_broadcast(pthread_cond_t *cond);
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t * mutex);
```