Operativni sistemi Januar 1 17.01.2017.

Napraviti u **/home/ispit1** direktorijum u skladu sa indeksom i tokom kojem pripadate. Na primer, student prvog toka sa indeksom 101/2015 treba da napravi folder **1_mi15101**, a student drugog toka sa indeksom 12/2015 treba da napravi folder **2_mi15012**. 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!*

1. Napisati program koji demonstrira upotrebu signala. Program u beskonacnoj petlji motri na emitovanje signala SIGUSR1, SIGUSR2 i SIGTERM. Za prva dva potrebno je instalirati rukovaoce signalima (eng. signal handlers) koji broje koliko su se puta ovi signali desili (uvećati odgovarajući brojač u rukovaocu). SIGTERM obraditi rukovaocem koji ispisuje koliko se puta SIGUSR1 desio i zatim koliko puta se desio SIGUSR2 (tim redom 2 broja) i završava program *exit code*-om 0. Dakle, ukoliko je program primio 2 signala SIGUSR1 i 5 signala SIGUSR2 izlaz iz programa je '**2 5'**, a *exit code* je 0.

2. Napisati program koji pokreće dete proces i u njemu komandu terminala *stat* za putanju prosleđenu kao argument komandne linije. Takođe, potrebno je preusmeriti standardni izlaz komande *stat* i obraditi ga tako da se u roditeljskom procesu ispise veličina fajla (jedan broj bez dodatnog ispisa). Na kraju ukoliko se komanda *stat* ne završi uspešno (proveriti *exit code*) ispisati '**Neuspeh**' iz roditeljskog procesa.

3. Napisati program koji koristi više niti da izračuna minimum matrice. Program kao argument komandne linije prima putanju do fajla gde se nalazi matrica. Prva dva broja u fajlu su broj redova matrice *N* i broj kolona matrice *M*, a nakon ova dva broja slede elementi matrice koji su realni brojevi (možete pretpostaviti da je ulaz ispravan). Potrebno je pokrenuti *N* niti tako da svaka nit obrađuje jedan red (računa minimum tog reda i ažurira globalni minimum ako je to potrebno). Koristiti muteks za sinhronizaciju. Ispisati vrednost minimuma matrice iz *main()* funkcije (jedan broj bez dodatnog teksta).

4. Napisati program koji kao argument komandne linije prima putanju do fajla i dva broja ***a*** i ***b***. Prvi broj predstavlja udaljenje od početka fajla, a drugi je broj bajtova. U slučaju da sekcija fajla [**a**, **a+b**] nije zaključana ispisati na standardni izlaz '*unlocked*', u slučaju da je zaključana da može da se čita, a ne može da se piše ispisati '*shared lock*' i na kraju ako je zaključana i za čitanje i za pisanje ispisati '*exclusive lock*'.

5. Napisati program koji kao argumente komandne linije prima putanje do objekta *deljene memorije.* Potrebno učitati strukturu:

```
typedef struct {
     sem_t inDataReady;
     float array[ARRAY_MAX];
     unsigned arrayLen;
} OsInputData;
```

i naći medijanu niza *array* (medijana je element sa indeksom *N/2* u nerastuće sortiranom nizu). Dodatno, pre bilo kakvog obrade, potrebno je sačekati na semafor *inDataReady* (pretpostaviti da je ispravno inicijalizovan). Izračunatu medijanu ispisati na standardni ulaz. NAPOMENA: Linkovati sa **-lrt**.

Pokretanje programa:	./1 /inmem	./1	./1 /nepostoji	./1 /somemem
Vrednosti *array:*	3.0, 1.0, 2.0, 5.0, 4.0	------	----------	7.0 -5.0 12.0 10.0
Standardni izlaz:	3.0	------	----------	10.0
Exit kod:	0	1	1	0

POSIX niti - dodatak

Sve funkcije za rad sa POSIX nitima vraćaju pozitivnu vrednost koda greške ako je do greške došlo, a nulu inače. 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);
```