

Operativni sistemi Jun 1 20.06.2017.

grupa A

Napraviti u **/home/ispit1** direktorijum u skladu sa indeksom, semestrom i asistentom kod koga slusate kurs. Na primer, student koji slusa kurs u prvom semestru kod Vlade, sa indeksom 101/2015, treba da napravi folder **v1_mi15101_a**, a student sa indeksom 12/2015 koji slusa kurs kod Ognjena u drugom semestru treba da napravi folder **o2_mi15012_a**. 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 greske mozete ispisivati sta god zelite. Strogo se drzite navedenih formata ispisa za standardni izlaz!

Zabranjeno je koriscenje system() funkcije! Kod prevoditi sa -std=c99 opcijom!

1. Napisati program koji za trenutak pokretanja ispisuje na standardni izlaz koji je dan u godini (npr. ako se program pokrene prvog januara ipisuje: **1**). **Ispis je samo jedan broj.**
2. Napisati program koji pokrece dete proces i u njemu komandu terminala ls -l za putanju do regularnog fajla prosledjenu kao argument komandne linije (ako putanja postoji fajl ce biti regularan). Takođe, potrebno je preusmeriti standardni izlaz komande ls -l i obraditi ga tako da se u **roditeljskom procesu** ispisu prava pristupa (npr. **-rwxr-xr-x**, prva kolona ispisa). Ukoliko se komanda ls -l ne završi uspešno (proveriti *exit code*) ispisati **'Neuspeh'** iz roditeljskog procesa (završiti sa exit code-om 0). Ukoliko nije prosledjen argument komandne linije završiti sa *exit code-om* 1. **Pomoc:** otvoriti *pipe* i izvršiti redirekciju u detetu tako da se *stdout* deteta preusmeri kroz *pipe* ka roditelju, a zatim nakon toga pozvati odgovarajući *exec* u detetu. Primer pokretanja: ./2 dir/1.txt .
3. Napisati program koji koristi vise niti da pomnozi 2 matrice dimenzija NxM i MxK. Sa standardnog ulaza se ucitavaju N i M, pa zatim NxM celih brojeva. Nakon toga ucitavaju se M i K, pa zatim MxK celih brojeva. Potrebno je pokrenuti NxK niti tako da svaka nit racuna jedan element rezultujuce matrice. Tokom izracunavanja, potrebno je cuvati maksimalni element rezultujuce matrice (**za sinhronizaciju koristiti muteks, nije dozvoljena staticka alokacija**). Iz *main()*-a ispisati rezultujucu matricu, **razdvojiti elemente belinom**, i kao poslednji broj ispisati i **maksimalni element**. Za matrice A i B:

A B

1 2 3 x 1 2 = **6 12** [maksimum = 30, broj niti koje treba pokrenuti 4 (N=2, M=3, K=2)]
4 5 6 1 2 **15 30**
 1 2

ispis je:

6 12

15 30

30

4. Napisati program koji kao argument komandne linije prima putanju do fajla. Fajl sadrzi tekst u kome se nalaze brojevi. Program treba da cita rec po rec (**maksimalna duzina reci 256**) i ukoliko je procitana rec broj, da se potruzi da ga zakljuca **za citanje**. Ukoliko je broj ne moze da se zakljuca, program normalno nastavlja dalje sa obradom. Ispisati na standardni izlaz broj uspesno zakljucanih brojeva (**ispis je samo jedan broj**). Proveriti broj argumenata komandne linije i da li fajl postoji (exit code 1 ako nesto od toga nije ispunjeno).
5. Napisati program koji kao argumente komandne linije prima ime objekta *deljene memorije*. Potrebno učitati strukturu (**ARRAY_MAX** je 1024):

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

i ispisati na **standardni izlaz** standardnu devijaciju niza. Dodatno, pre bilo kakvog obrade, potrebno je sacekati na semafor *inDataReady* (pretpostaviti da je ispravno inicijalizovan). Standardna devijacija se racuna kao:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}, \text{ where } \mu = \frac{1}{N} \sum_{i=1}^N x_i.$$

<u>Pokretanje programa:</u>	./1 /inmem	./1	./1 /nepostoji	./1 /somemem
<u>Vrednosti array:</u>	3.0, 1.0, 2.0, 5.0, 4.0	-----	-----	7.0, -5.0, 12.0, 10.0
<u>Standardni izlaz:</u>	1.58114	-----	-----	7.61577
<u>Exit kod:</u>	0	1	1	0

POSIX niti - dodatak

Sve funkcije za rad sa POSIX nitima vraćaju pozitivnu vrednost koda greške 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);
```