**Reseni rokovi iz OS**

**Januar 1 2017:**

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 SIGUSRI 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.

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <signal.h>

#include <stdint.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

int sigusr1Caught = 0;

int sigusr2Caught = 0;

int sigtermCaught = 0;

void sig\_handler(int signum)

{

switch(signum){

case SIGUSR1:

sigusr1Caught = 1;

break;

case SIGUSR2:

sigusr2Caught = 1;

break;

case SIGTERM:

sigtermCaught = 1;

break;

}

}

int main(int argc, char \*\*argv)

{

check(argc == 1, "Usage: ./1");

int numOfUsr1 = 0;

int numOfUsr2 = 0;

//printf("PID: %jd\n", (intmax\_t)getpid());

do {

check(signal(SIGUSR1, sig\_handler) != SIG\_ERR, "signal usr1");

check(signal(SIGUSR2, sig\_handler) != SIG\_ERR, "signal usr2");

check(signal(SIGTERM, sig\_handler) != SIG\_ERR, "signal term");

pause();

if(sigusr1Caught) {

sigusr1Caught = 0;

numOfUsr1++;

}

if(sigusr2Caught) {

sigusr2Caught = 0;

numOfUsr2++;

}

} while(!sigtermCaught);

printf("%d %d", numOfUsr1, numOfUsr2);

return 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.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

#include <string.h>

#include <ctype.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define RD\_END (0)

#define WR\_END (1)

int main(int argc, char \*\*argv)

{

check(argc == 2, "Usage: ./2 pathToFile");

int cld2par[2];

check(pipe(cld2par) != -1, "pipe");

pid\_t childPid = fork();

check(childPid != -1, "fork");

if(childPid > 0) // parent

{

close(cld2par[WR\_END]);

FILE \*f = fdopen(cld2par[RD\_END], "r");

check(f != NULL, "fdopen");

char \*line = NULL;

size\_t lineLen = 0;

while(getline(&line, &lineLen, f) != -1)

{

char \*sub;

char \*needle = "Size: ";

sub = strstr(line, needle);

if(sub != NULL)

{

size\_t size = strlen(needle);

sub += size;

while(!isspace((unsigned char)\*sub))

{

printf("%c", \*sub);

sub++;

}

}

}

close(cld2par[RD\_END]);

}

else //child

{

close(cld2par[RD\_END]);

check(dup2(cld2par[WR\_END], STDOUT\_FILENO) != -1, "dup2");

check(execlp("stat", "stat", argv[1], NULL) != -1, "execlp");

close(cld2par[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

if(WIFEXITED(status) && (WEXITSTATUS(status) == EXIT\_SUCCESS))

{

}

else

{

printf("Neuspeh");

}

return 0;

}

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).

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <errno.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define check\_pthread(err, msg) \

do { \

int \_err = err; \

if(\_err > 0) { \

errno = \_err; \

check(0, msg); \

} \

} while(0)

typedef struct{

int idx;

} data\_t;

int m, n;

double \*\*matrica;

int init = 0;

double globalMin;

pthread\_mutex\_t globalLock = PTHREAD\_MUTEX\_INITIALIZER;

void \*threadFunc(void \* arg)

{

data\_t \*data = (data\_t \*)arg;

double localMin = matrica[data->idx][0];

for(int i = 0; i < m; i++)

if(matrica[data->idx][i] < localMin)

localMin = matrica[data->idx][i];

check\_pthread(pthread\_mutex\_lock(&globalLock), "lock");

if(init)

{

if(localMin < globalMin)

{

globalMin = localMin;

}

}

else

{

globalMin = localMin;

init = 1;

}

check\_pthread(pthread\_mutex\_unlock(&globalLock), "unlock");

return NULL;

}

int main(int argc, char \*\*argv)

{

check(argc == 2, "Usage: ./3 pathToFile");

FILE \*f = fopen(argv[1], "r");

check(f != NULL, "fopen");

fscanf(f, "%d %d", &n, &m);

matrica = malloc(n \* sizeof(double\*));

check(matrica != NULL, "malloc");

for(int i = 0; i < n; i++)

{

matrica[i] = malloc(m \* sizeof(double));

check(matrica[i] != NULL, "malloc");

}

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++)

fscanf(f, "%lf", &matrica[i][j]);

fclose(f);

pthread\_t \*tids = malloc(sizeof(pthread\_t) \* n);

check(tids != NULL, "malloc");

data\_t \*args = malloc(sizeof(data\_t) \* n);

check(args != NULL, "malloc");

for(int i = 0; i < n; i++)

{

args[i].idx = i;

check\_pthread(pthread\_create(&tids[i], NULL, threadFunc, &args[i]), "pthread\_create");

}

for(int i = 0; i < n; i++)

{

check\_pthread(pthread\_join(tids[i], NULL), "pthread\_join");

}

printf("%lf", globalMin);

for(int i = 0; i < n; i++)

free(matrica[i]);

free(matrica);

free(tids);

free(args);

return 0;

}

4. Napisati program koji kao argument komandne linije prima putanju do fajla i dva broja a i b. Prvi broj predstavlja údaljenje 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'.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

int main(int argc, char \*\*argv)

{

check(argc == 4, "Usage: ./4 pathToFile a b");

int read = 0;

int write = 0;

struct flock lock;

lock.l\_type = F\_RDLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = atoi(argv[2]);

lock.l\_len = atoi(argv[3]);

int fd = open(argv[1], O\_RDWR);

check(fd != -1, "open");

check(fcntl(fd, F\_GETLK, &lock) != -1, "fcntl");

if(lock.l\_type == F\_UNLCK)

read = 1;

lock.l\_type = F\_WRLCK;

check(fcntl(fd, F\_GETLK, &lock) != -1, "fcntl");

if(lock.l\_type == F\_UNLCK)

write = 1;

if(read && write)

printf("unlocked");

else if(read && !write)

printf("shared lock");

else

printf("exclusive lock");

close(fd);

return 0;

}

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.

#include <stdio.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <semaphore.h>

#include <sys/types.h>

#include <unistd.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define ARRAY\_MAX (1024)

typedef struct {

sem\_t inDataReady;

float array[ARRAY\_MAX];

unsigned arrayLen;

} osInputData;

void \*getMemoryBlock(char \*fpath, unsigned \*size);

int cmpFl(const void \*a, const void \*b);

int main(int argc, char \*\*argv)

{

check(argc == 2, "Usage: ./5 pathToMem");

unsigned size = 0;

osInputData \*sharedMem = getMemoryBlock(argv[1], &size);

check(sem\_wait(&(sharedMem->inDataReady)) != -1, "sem wait");

qsort(&sharedMem->array, sharedMem->arrayLen, sizeof(float), cmpFl);

if(sharedMem->arrayLen % 2 == 1)

printf("%f\n", sharedMem->array[sharedMem->arrayLen / 2]);

else

printf("%f\n", sharedMem->array[sharedMem->arrayLen / 2 - 1]);

check(munmap(sharedMem, size) != -1, "munmap");

return 0;

}

void \*getMemoryBlock(char \*fpath, unsigned \*size)

{

int memFd = shm\_open(fpath, O\_RDWR, 0600);

check(memFd != -1, "shm\_open");

struct stat fInfo;

check(fstat(memFd, &fInfo) != -1, "fstat");

\*size = fInfo.st\_size;

void \*addr;

check((addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0)) != MAP\_FAILED, "mmap");

close(memFd);

return addr;

}

int cmpFl(const void \*a, const void \*b)

{

if(\*(float\*)a > \*(float\*)b)

return -1;

else if(\*(float\*)a < \*(float\*)b)

return 1;

else

return 0;

}

**Jun 1 2017:**

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 Is -1 ne završi uspešno (proveriti exit code) ispisati 'Neuspeh' iz roditeljskog procesa (zavrsiti sa exit code-om 0). Ukoliko nije prosledjen argument komandne linije zavrsiti sa exit code-om 1. Pomoc: otvoriti pipe i izvrsiti redirekciju u detetu tako da se stdout deteta preusmeri kroz pipe ka roditelju, a zatim nakon toga pozvati odgovarajuci exec u detetu. Primer pokretanja: /2 dir/1.txt.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

#include <fcntl.h>

#include <ctype.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define RD\_END (0)

#define WR\_END (1)

#define MAX\_SIZE (1024)

int main(int argc, char \*\*argv)

{

check(argc == 2, "Usage: ./2 pathToFile");

int cld2par[2];

check(pipe(cld2par) != -1, "pipe");

pid\_t childPid = fork();

check(childPid != -1, "fork");

if(childPid > 0) // parent

{

close(cld2par[WR\_END]);

FILE \*f = fdopen(cld2par[RD\_END], "r");

check(f != NULL, "fdopen");

char c;

while((c = fgetc(f)) != EOF)

{

if(isspace(c))

break;

printf("%c", c);

}

close(cld2par[RD\_END]);

}

else // child

{

close(cld2par[RD\_END]);

check(dup2(cld2par[WR\_END], STDOUT\_FILENO) != -1, "dup2");

check(execlp("ls", "ls", "-l", argv[1], NULL) != -1, "execlp");

close(cld2par[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

if(WIFEXITED(status) && (WEXITSTATUS(status) == EXIT\_SUCCESS))

{

}

else

{

printf("Neuspeh");

}

return 0;

}

3. Napisati program koji koristi vise niti da pomnozi 2 matrice dimenzija NxM i MxK. Sa standardnog ulaza se ucitavaju Ni M, pa zatim NxM celih brojeva. Nakon toga ucitavaju se Mi

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

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <errno.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define check\_pthread(err, msg) \

do { \

int \_err = err; \

if(\_err > 0) { \

errno = \_err; \

check(0, msg); \

} \

} while(0)

int n, m, m1, k;

int \*\*matricaA;

int \*\*matricaB;

int \*\*matricaC;

pthread\_mutex\_t globalLock = PTHREAD\_MUTEX\_INITIALIZER;

int globalMax;

int init = 0;

typedef struct {

int idx;

} data\_t;

void \*threadFunc(void \*arg)

{

data\_t\* data = (data\_t \*)arg;

int p = data->idx / k;

int d = data->idx % k;

int res = 0;

for(int i = 0; i < m; i++)

{

res += matricaA[p][i] \* matricaB[i][d];

}

matricaC[p][d] = res;

int localMax = matricaC[p][d];

check\_pthread(pthread\_mutex\_lock(&globalLock), "lock");

if(init)

{

if(localMax > globalMax)

globalMax = localMax;

}

else

{

globalMax = localMax;

init = 1;

}

check\_pthread(pthread\_mutex\_unlock(&globalLock), "unlock");

return NULL;

}

int main(int argc, char \*\*argv)

{

check(argc == 1, "Usage: ./3");

scanf("%d %d", &n, &m);

matricaA = malloc(n \* sizeof(int\*));

check(matricaA != NULL, "malloc");

for(int i = 0; i < n; i++)

{

matricaA[i] = malloc(m \* sizeof(int));

check(matricaA[i] != NULL, "malloc");

}

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++)

scanf("%d", &matricaA[i][j]);

scanf("%d %d", &m1, &k);

check(m == m1, "not equal");

matricaB = malloc(m \* sizeof(int\*));

check(matricaB != NULL, "malloc");

for(int i = 0; i < m; i++)

{

matricaB[i] = malloc(k \* sizeof(int));

check(matricaB[i] != NULL, "malloc");

}

for(int i = 0; i < m; i++)

for(int j = 0; j < k; j++)

scanf("%d", &matricaB[i][j]);

matricaC = malloc(n \* sizeof(int\*));

check(matricaC != NULL, "malloc");

for(int i = 0; i < n; i++)

{

matricaC[i] = malloc(k \* sizeof(int));

check(matricaC[i] != NULL, "malloc");

}

pthread\_t\* tids = malloc(n \* k \* sizeof(pthread\_t));

check(tids != NULL, "malloc");

data\_t \*args = malloc(n \* k \* sizeof(data\_t));

check(args != NULL, "malloc");

for(int i = 0; i < n \* k; i++)

{

args[i].idx = i;

check\_pthread(pthread\_create(&tids[i], NULL, threadFunc, &args[i]), "pthread\_create");

}

for(int i = 0; i < n \* k; i++)

{

check\_pthread(pthread\_join(tids[i], NULL), "pthread\_join");

}

for(int i = 0; i < n; i++)

{

for(int j = 0; j < k; j++)

printf("%d ", matricaC[i][j]);

printf("\n");

}

printf("%d", globalMax);

for(int i = 0; i < n; i++)

free(matricaA[i]);

free(matricaA);

for(int i = 0; i < m; i++)

free(matricaB[i]);

free(matricaB);

for(int i = 0; i < n; i++)

free(matricaC[i]);

free(matricaC);

free(tids);

free(args);

return 0;

}

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) 1 ukoliko je procitana rec broj, da se potrudi 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 fail postoji (exit code l'ako nesto od toga nije ispunjeno).

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <errno.h>

#include <string.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc == 2, "Usage: ./4 pathToFile");

int cnt = 0;

FILE \*f = fopen(argv[1], "r");

check(f != NULL, "fopen");

int fd = fileno(f);

check(fd != -1, "fileno");

struct flock lock;

char buffer[MAX\_SIZE];

while(!feof(f))

{

char \*err = buffer;

memset(buffer, 0, MAX\_SIZE);

fscanf(f, "%s", buffer);

int num = strtol(buffer, &err, 10);

if(err == buffer || \*err != '\0')

continue;

int len = strlen(buffer);

lock.l\_type = F\_RDLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = ftell(f);

lock.l\_len = -len;

int res = fcntl(fd, F\_SETLK, &lock);

if(res == -1)

{

if(errno != EACCES && errno != EAGAIN)

check(0, "fcntl");

}

else

{

cnt++;

}

}

lock.l\_type = F\_UNLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = 0;

lock.l\_len = 0;

check(fcntl(fd, F\_SETLK, &lock) != -1, "unlocking failed");

printf("%d\n", cnt);

fclose(f);

return 0;

}

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

#define \_XOPEN\_SOURCE 700

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <stdbool.h>

#include <errno.h>

#include <sys/mman.h>

#include <semaphore.h>

#include <math.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define MAX\_SIZE (1024)

typedef struct {

sem\_t inDataReady;

float array[MAX\_SIZE];

int arrayLen;

} osInputData;

void\* osGetMemoryBlock(const char\* path, int\* size);

int main(int argc, char\*\* argv) {

check(argc == 2, "./5 pathToFile");

int size = 0;

osInputData\* data = osGetMemoryBlock(argv[1], &size);

check(sem\_wait(&(data->inDataReady)) != -1, "sem wait failed");

float sum = 0;

int i = 0;

for (i = 0; i < data->arrayLen; i++) {

sum += data->array[i];

}

sum /= data->arrayLen;

float stdDev = 0;

for (i = 0; i < data->arrayLen; i++)

{

stdDev += (data->array[i] - sum)\*(data->array[i] - sum);

}

stdDev = sqrt(stdDev / data->arrayLen);

printf("%f\n", stdDev);

check(munmap(data, size) != -1, "munmap failed");

exit(EXIT\_SUCCESS);

}

void\* osGetMemoryBlock(const char\* path, int\* size) {

int memFd = shm\_open(path, O\_RDWR, 0600);

check(memFd != -1, "shmopen failed");

struct stat fInfo;

check(fstat(memFd, &fInfo) != -1, "stat failed");

\*size = fInfo.st\_size;

void\* addr;

check((addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0)) != MAP\_FAILED, "mmap failed");

return addr;

}

**Jun2 2017:**

1. Napisati program koji za trenutak pokretanja ispisuje na standardni izlaz koji je broj nedelje u godini (nedelje se broje pocevsi od prvog ponedeljka u godini). Ispis je samo jedan broj.

2. Napisati program koji u dete procesu pokrece komandu find type f i prosledjuje rezultat roditelju. Roditeljski proces treba da ispise broj fajlova cija ekstenzija odgovara ekstenziji koja se dobija kao argument komandne linije. Pokretanje: /2.txt. Pomoc: otvoriti pipe i izvrsiti redirekciju u detetu tako da se stdout deteta preusmeri kroz pipe ka roditelju, a zatim nakon toga pozvati odgovarajuci exec u detetu; u roditelju citati iz pipe-a liniju po liniju.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define RD\_END (0)

#define WR\_END (1)

int main(int argc, char \*\*argv)

{

check(argc == 2, "./1 pathToFile");

int cld2par[2];

check(pipe(cld2par) != -1, "pipe");

int childPid = fork();

check(childPid != -1, "fork");

char \*ext;

ext = strrchr(argv[1], '.');

check(ext != NULL, "strrchr");

strncat(ext, "\n", 1);

int cnt = 0;

if(childPid > 0) //parent

{

close(cld2par[WR\_END]);

FILE \*f = fdopen(cld2par[RD\_END], "r");

check(f != NULL, "fdopen");

size\_t lineLen = 0;

char \*line = NULL;

while(getline(&line, &lineLen, f) != -1)

{

char \*sub;

sub = strrchr(line, '.');

if(!strcmp(sub, ext))

cnt++;

}

printf("%d", cnt);

free(line);

fclose(f);

close(cld2par[RD\_END]);

}

else //child

{

close(cld2par[RD\_END]);

check(dup2(cld2par[WR\_END], STDOUT\_FILENO) != -1, "dup2");

check(execlp("find", "find", ".", "-type", "f", NULL) != -1, "execlp");

close(cld2par[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

return 0;

}

3. Napisati program koji kad argument komandne linije prima putanju do fajla i dve reci. U navedenom fajlu program treba da zameni sva pojavljivanja prve reci sa drugom reci (pretpostaviti da su reci iste duzine). Pre nego sto prepise staru rec, program treba da je zakljuca. Ukoliko je druge proces vec zakljucao neko pojavljivanje reci, nastaviti normalno sa obradom bez zamene reci. Dodatno, potrebno je ispisati broj reci koje nisu zamenjene (zbog toga sto je drugi proces drzao katanac). Maksimalna duzina reci u fajlu je 256.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <errno.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc == 4, "./3 pathToFile str1 str2");

FILE \*f = fopen(argv[1], "r+");

check(f != NULL, "fopen");

int fd = fileno(f);

check(fd != -1, "open");

struct flock lock;

char buf[MAX\_SIZE];

unsigned int cnt = 0;

while(!feof(f))

{

memset(buf, 0, MAX\_SIZE);

fscanf(f, "%s", buf);

int len = strlen(buf);

if(strcmp(argv[2], buf))

continue;

lock.l\_type = F\_WRLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = ftell(f);

lock.l\_len = -len;

int res = fcntl(fd, F\_SETLK, &lock);

if(res == -1)

{

if (errno != EACCES && errno != EAGAIN)

check(0, "lock failed");

cnt++;

}

else

{

check(fseek(f, -len, SEEK\_CUR) != -1, "lseek failed");

fprintf(f, "%s", argv[3]);

}

}

lock.l\_type = F\_UNLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = 0;

lock.l\_len = 0;

check(fcntl(fd, F\_SETLK, &lock) != -1, "fcntl");

printf("%u", cnt);

fclose(f);

return 0;

}

4. Napisati program koji kao argumente komandne linije prima putanje do FIFO fajlova. Program istovremeno za sve fajlove ceka na dogadjaj da je kroz neki FIFO stigao podatak koriscenjem poll interfejsa. Kroz FIFO fanove pristizu samo celi brojevi, cita se sve dok se ne zatvore svi FIFO fajlovi na strani koja pise u FIFO. Na standardni izlaz ispisati maksimalni procitan broj i naziv FIFO fajla (ne putanju nego NAZIV).

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <poll.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

int main(int argc, char \*\*argv)

{

check(argc > 2, "./1 pathToFifo pathToFifo ...");

int numFifos = argc - 1;

struct pollfd \*fileMonitors = calloc(numFifos, sizeof(struct pollfd));

check(fileMonitors != NULL, "calloc");

FILE \*\*streams = malloc(numFifos \* sizeof(FILE\*));

check(streams != NULL, "malloc");

for(int i = 0; i < numFifos; i++)

{

fileMonitors[i].fd = open(argv[i+1], O\_RDONLY | O\_NONBLOCK);

check(fileMonitors[i].fd != -1, "open");

fileMonitors[i].events = POLLIN | POLLHUP | POLLERR;

streams[i] = fdopen(fileMonitors[i].fd, "r");

check(streams[i] != NULL ,"fdopen");

}

char \*file = NULL;

int read\_num;

int max\_num;

int init = 0;

int numActive = numFifos;

while(numActive)

{

int events = poll(fileMonitors, numFifos, -1);

check(events != -1, "poll");

for(int i = 0; i < numFifos; i++)

{

if(fileMonitors[i].revents & POLLIN)

{

while(fscanf(streams[i], "%d", &read\_num) == 1)

{

if(init)

{

if(read\_num > max\_num)

{

max\_num = read\_num;

file = argv[i + 1];

}

}

else

{

max\_num = read\_num;

file = argv[i + 1];

init = 1;

}

}

fileMonitors[i].revents = 0;

}

else if(fileMonitors[i].revents & (POLLHUP | POLLERR))

{

fclose(streams[i]);

fileMonitors[i].fd = -1;

fileMonitors[i].events = 0;

fileMonitors[i].revents = 0;

numActive--;

}

}

}

char \*name = strrchr(file, '/');

printf("%d %s", max\_num, name != NULL ? (name + 1) : file);

free(fileMonitors);

free(streams);

return 0;

}

5. Napisati program koji kao argumente komandne linije prima ime objekta deljene memorije. Potrebno ucitati strukturu (ARRAY\_MAX je 1024):

typedef struct {

sem\_t inDataReady;

int array[ARRAY\_MAX];

usigned arrayLen;

}osInputData;

I ispisati na stadnardni izlaz brojeve ciji binarni zapis ima bar 4 jedinice u sebi. Dodatno, pre bilo kakve obrade, potrebno je sacekati na semafor inDataReady (pretpostaviti da je ispravno inicijalizovan). Ne postavljati semafor nakon obrade!

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <semaphore.h>

#define check(expr, msg) \

do{ \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define ARRAY\_MAX (1024)

typedef struct {

sem\_t inDataReady;

int array[ARRAY\_MAX];

unsigned arrayLen;

} osInputData;

void \*getMemBlock(const char \*path, unsigned \*size)

{

int memFd = shm\_open(path, O\_RDWR, 0600);

check(memFd != -1, "shm\_open");

struct stat fInfo;

check(fstat(memFd, &fInfo) != -1, "fstat");

\*size = fInfo.st\_size;

void \*addr;

addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0);

check(addr != MAP\_FAILED, "mmap failed");

close(memFd);

return addr;

}

int main(int argc, char \*\*argv)

{

check(argc == 2, "./1 pathToFile");

unsigned size = 0;

osInputData \*data = getMemBlock(argv[1], &size);

check(sem\_wait(&data->inDataReady) != -1, "sem\_wait");

for(unsigned i = 0; i < data->arrayLen; i++)

{

int num = data->array[i];

int cnt = 0;

int cpy = num;

do

{

if(cpy % 2 == 1)

cnt++;

cpy /= 2;

}while(cpy != 0);

if(cnt >= 4)

printf("%d ", num);

}

check(munmap(data, size) != -1, "munmap");

//check(shm\_unlink(argv[1]) != -1, "shm\_unlink");

return 0;

}

**Septembar1 2017:**

1. Napisati program koji u petlji ucitava rec ne duzu od 63 karaktera sa standardnog ulaza. Nakon ucitavanja reci program ceka da mu se posalje signal SIGUSR1, SIGUSR2 ili SIGTERM (cekanje na signal je u istoj petlji kao citanje stringa). Ukoliko program primi SIGUSR1 na standardni izlaz ispisuje ucitanu rec naopako, ukoliko primi SIGUSR2 ispisuje ucitanu rec u kojoj su mala slova zamenjena velikim slovima, ukoliko primi SIGTERM program prekida sa izvrsavanjem (exit code 0). String obradjivati unutar petlje (u signal handler-u samo postaviti flag koji govori koja operacija treba da se izvrsi). Primer pokretanja (obavezno kompajliranje sa -std=c99).

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <unistd.h>

#include <string.h>

#include <stdint.h>

#include <ctype.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define MAX\_SIZE (64)

int \_sigusr1 = 0;

int \_sigusr2 = 0;

int terminate = 0;

void sigHandler(int signum);

int main(int argc, char \*\*argv)

{

check(argc == 1, "./1");

//printf("%ld\n", (intmax\_t)getpid());

char str[MAX\_SIZE];

do

{

check(signal(SIGUSR1, sigHandler) != SIG\_ERR, "signal");

check(signal(SIGUSR2, sigHandler) != SIG\_ERR, "signal");

check(signal(SIGTERM, sigHandler) != SIG\_ERR, "signal");

pause();

if(terminate)

break;

scanf("%s", str);

if(\_sigusr1)

{

\_sigusr1 = 0;

int len = strlen(str);

int end = len - 1;

char res[MAX\_SIZE];

int i;

for(i = 0; i < len; i++)

{

res[i] = str[end];

end--;

}

res[i] = '\0';

printf("%s\n", res);

}

else if(\_sigusr2)

{

\_sigusr2 = 0;

int len = strlen(str);

for(int i = 0; i < len; i++)

{

str[i] = toupper(str[i]);

}

printf("%s\n", str);

}

} while(1);

return 0;

}

void sigHandler(int signum)

{

switch(signum)

{

case SIGUSR1:

\_sigusr1 = 1;

break;

case SIGUSR2:

\_sigusr2 = 1;

break;

case SIGTERM:

terminate = 1;

break;

}

}

2. Sa standardnog ulaza se ucitava broj cvorova grafa |V|, a zatim i graf predstavljen matricom povezanosti M dimezija |V|x|V|. Ukoliko postoji grana od cvora i do cvora j onda je M[i][j] = 1, a nula inace. Pokrenuti |V| niti i u svakoj izracunati ulazni stepen odgovarajuceg cvora (u nultoj niti za cvor i=0 itd.). Ispisati indeks cvora sa najvecim ulaznim stepenom iz main funkcije. Koristiti muteks za sinhronizaciju. Primer pokretanja:

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <errno.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define check\_pthread(err, msg) \

do { \

int \_err = err; \

if(\_err > 0) { \

errno = \_err; \

check(0, msg); \

} \

} while(0)

typedef struct {

int idx;

} data\_t;

int maxGl;

int maxTh;

int init = 0;

int \*\*matrica;

int v;

pthread\_mutex\_t globalMutex = PTHREAD\_MUTEX\_INITIALIZER;

void \*threadFunc(void \*arg)

{

data\_t \*data = (data\_t\*) arg;

int in = 0;

for(int i = 0; i < v; i++)

in += matrica[i][data->idx];

check\_pthread(pthread\_mutex\_lock(&globalMutex), "lock");

if(init)

{

if(in > maxGl)

{

maxGl = in;

maxTh = data->idx;

}

}

else

{

init = 1;

maxGl = in;

maxTh = data->idx;

}

check\_pthread(pthread\_mutex\_unlock(&globalMutex), "unlock");

return NULL;

}

int main(int argc, char \*\*argv)

{

check(argc == 1, "./2");

scanf("%d", &v);

matrica = malloc(v \* sizeof(int\*));

check(matrica != NULL, "malloc");

for(int i = 0; i < v; i++)

{

matrica[i] = malloc(v \* sizeof(int));

check(matrica[i] != NULL, "malloc");

}

for(int i = 0; i < v; i++)

for(int j = 0; j < v; j++)

scanf("%d", &matrica[i][j]);

pthread\_t \*tids = malloc(v \* sizeof(pthread\_t));

check(tids != NULL, "malloc");

data\_t \*args = malloc(v \* sizeof(data\_t));

check(args != NULL, "malloc");

for(int i = 0; i < v; i++)

{

args[i].idx = i;

check\_pthread(pthread\_create(&tids[i], NULL, threadFunc, &args[i]), "pthread\_create");

}

for(int i = 0; i < v; i++)

check\_pthread(pthread\_join(tids[i], NULL), "pthread\_join");

printf("%d", maxTh);

for(int i = 0; i < v; i++)

free(matrica[i]);

free(matrica);

free(tids);

free(args);

return 0;

}

3. Napisati program koji kao argumente komandne linije prima relativne putanje do FIFO fajlova. Koriscenjem epoll API-ja program istovremeno motri ove fajlove cekajuci na dogadjaj citanja.

Program cita realne brojeve iz FIFO fajlova sve dok ima brojeva I ispisuje relativnu putanju do FIFO fajla ciji je zbir procitanih brojeva najveci.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <sys/epoll.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <unistd.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define MAX\_EVENTS (8)

#define WAIT\_TIME (-1)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc >= 3 , "./3 fifo1 fifo2 ...");

int numFifos = argc - 1;

int epollFd = epoll\_create(numFifos);

check(epollFd != -1, "epoll\_create");

struct epoll\_event \*events = malloc(numFifos \* sizeof(struct epoll\_event));

check(events != NULL, "malloc");

int \*fds = malloc(numFifos \* sizeof(int));

check(fds != NULL, "malloc");

double \*sums = malloc(numFifos \* sizeof(double));

check(sums != NULL, "malloc");

for(int i = 0; i < numFifos; i++)

{

int fd = open(argv[i+1], O\_RDONLY|O\_NONBLOCK);

check(fd != -1, "open");

events[i].events = EPOLLIN | EPOLLHUP | EPOLLERR;

events[i].data.fd = fd;

fds[i] = events[i].data.fd;

check(epoll\_ctl(epollFd, EPOLL\_CTL\_ADD, fd, &events[i]) != -1, "epoll\_ctl");

}

struct epoll\_event readyFds[MAX\_EVENTS];

int activeFifos = numFifos;

while(activeFifos)

{

int numReadyEvents;

check((numReadyEvents = epoll\_wait(epollFd, readyFds, MAX\_EVENTS, WAIT\_TIME)) != -1, "epoll\_wait");

for(int i = 0; i < numReadyEvents; i++)

{

if(readyFds[i].events & EPOLLIN)

{

double num;

double sum = 0;

FILE \*f = fdopen(readyFds[i].data.fd, "r");

check(f != NULL, "fdopen");

while(fscanf(f, "%lf", &num) == 1)

{

sum+= num;

}

for(int j = 0 ; j < numFifos; j++)

{

if(fds[j] == readyFds[i].data.fd)

{

sums[j] = sum;

}

}

}

else if(readyFds[i].events & (EPOLLHUP|EPOLLERR))

{

close(readyFds[i].data.fd);

activeFifos--;

}

}

}

double max\_sum = sums[0];

int k = 0;

for(int i = 0; i < numFifos; i++)

{

if(sums[i] > max\_sum)

{

k = i;

max\_sum = sums[i];

}

}

char \*line = strrchr(argv[k + 1], '/');

char \*path = line != NULL ? line + 1 : argv[k + 1];

printf("%s\n", path);

close(epollFd);

free(events);

return 0;

}

4. Napisati program koji kao argument komandne linije prima broj sekundi od Epohe i ispisuje naziv dana u nedelji na srpskom jeziku za slovima.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

int main(int argc, char \*\*argv)

{

check(argc == 2, "./2 sec");

long sec = atol(argv[1]);

struct tm \*local = localtime(&sec);

int weekDay = local->tm\_wday;

switch(weekDay) {

case 0:

printf("nedelja");

break;

case 1:

printf("ponedeljak");

break;

case 2:

printf("utorak");

break;

case 3:

printf("sreda");

break;

case 4:

printf("cetvrtak");

break;

case 5:

printf("petak");

break;

case 6:

printf("subota");

break;

default:

check(0, "switch");

break;

}

return 0;

}

5. Napisati program koji kao argument komandne linije prima broj sekundi od Epohe i ispisuje naziv dana u nedelji na srpskom jeziku za slovima. Primer pokretanja: prosledjeni broj sekundi. Naziv ispisati malim

Napisati program koji kao argumente komandne linije prima ime objekta deljene memorije. Potrebno ucitati strukturu (ARRAY\_MAX je

typedef struct { sem\_t inDataReady; int array[ARRAY\_MAX]; unsigned arrayLen; } OsInputData;

i ispisati na standardni izlaz brojeve koji su stepeni trojke. Dodatno, pre bilo kakve obrade, potrebno je sacekati na semafor inDataReady (pretpostaviti da je ispravno inicijalizovan).

#include <stdio.h>

#include <stdlib.h>

#include <semaphore.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/types.h>

#include <unistd.h>

#define check(expr, msg) \

do { \

if(!(expr)) { \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

} while(0)

#define ARRAY\_MAX (1024)

typedef struct {

sem\_t inDAtaReady;

int array[ARRAY\_MAX];

unsigned arrayLen;

} osInputData;

void \*getMemBlock(const char \*path, unsigned \*size)

{

int memFd = shm\_open(path, O\_RDWR, 0600);

check(memFd != -1, "shm\_open");

struct stat fInfo;

check(fstat(memFd, &fInfo) != -1, "fstat");

\*size = fInfo.st\_size;

void \*addr;

check((addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0)) != MAP\_FAILED, "mmap");

close(memFd);

return addr;

}

int main(int argc, char \*\*argv)

{

check(argc == 2, "./5 mem");

unsigned size = 0;

osInputData \*data = getMemBlock(argv[1], &size);

check(sem\_wait(&(data->inDAtaReady)) != -1, "sem\_wait");

for(unsigned i = 0; i < data->arrayLen; i++)

{

int num = data->array[i];

if(num == 1)

printf("%d ", num);

else

{

int cpy = num;

int jeste = 1;

do{

if(cpy / 3 == 1 && cpy % 3 == 0)

break;

if(cpy % 3 != 0)

{

jeste = 0;

break;

}

cpy /= 3;

} while(cpy != 0);

if(jeste)

printf("%d ", num);

}

}

check(munmap(data, size) != -1, "munmap failed");

return 0;

}

**Januar1 2018:**

1. Napisati program koji za broj sekundi od Epohe, prosledjen kao argument komandne linije, ispisuje naziv meseca koji odgovora tom trenutku. Nazivi meseca su srpski nazivi koji pocinju malim slovom (npr. Januar, februar…).

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define check(expr, msg) \

do { \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

int main(int argc, char \*\*argv)

{

check(argc == 2, "argc");

long seconds = atol(argv[1]);

struct tm \*time = localtime(&seconds);

check(time != NULL, "localtime");

int month = time->tm\_mon;

switch(month)

{

case 0:

printf("januar");

break;

case 1:

printf("februar");

break;

case 2:

printf("mart");

break;

case 3:

printf("april");

break;

case 4:

printf("maj");

break;

case 5:

printf("jun");

break;

case 6:

printf("jul");

break;

case 7:

printf("avgust");

break;

case 8:

printf("septembar");

break;

case 9:

printf("oktobar");

break;

case 10:

printf("novembar");

break;

case 11:

printf("decembar");

break;

}

return 0;

}

2. Napisati program koji kao argument komandne linije prima putanju do fajla, sekvencu karaktera W I broj niti K koje treba pokrenuti. Za fajl velicine N svaka nit broji pojavljivanja W u jednoj sekciji fajla I dodaje taj broj na globalnu sumu. Nit sa indeksom 0 u sekciji [0, N/K), nit sa indeksom 1 u sekciji [N/k, 2\*N/K) itd. Iz main funkcije ispisati ukupan broj pojavljivanja W u fajlu. Pretpostaviti da se W ne moze naci na granici izmedju sekcija I d aje N deljivo sa K.

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <pthread.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/stat.h>

#include <string.h>

#include <stdatomic.h>

#include <fcntl.h>

#define check(expr, msg) \

do { \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define check\_thread(err, msg) \

do{ \

int \_err = err; \

if(\_err > 0){ \

errno = \_err; \

check(0, msg); \

} \

}while(0)

#define MAX\_SIZE (256)

typedef struct yes{

char \*file;

int start;

int end;

}inData;

char \*W;

int K, N;

atomic\_int globalSum;

void \*threadFunc(void \*arg);

int main(int argc, char \*\*argv)

{

check(argc == 4, "argc");

atomic\_init(&globalSum, 0);

W = argv[2];

K = atoi(argv[3]);

struct stat fInfo;

check(stat(argv[1], &fInfo) != -1, "stat");

N = fInfo.st\_size;

pthread\_t \*tids = malloc(K \* sizeof(pthread\_t));

check(tids != NULL, "malloc");

inData \*args = malloc(K \* sizeof(inData));

check(args != NULL, "malloc");

for(int i = 0; i < K; i++)

{

args[i].file = argv[1];

args[i].start = i \* N/K;

args[i].end = i \* N/K + N/K;

check\_thread(pthread\_create(&tids[i], NULL, threadFunc, &args[i]), "pthread\_create");

}

for(int i = 0; i < K; i++)

{

check\_thread(pthread\_join(tids[i], NULL), "pthread\_join");

}

printf("%d\n", atomic\_load(&globalSum));

return 0;

}

void \*threadFunc(void \*arg)

{

inData \*data = (inData\*)arg;

char \*file = data->file;

int s = data->start;

int e = data->end;

int count = 0;

FILE \*f = fopen(file, "r");

check(f != NULL, "fopen");

check(fseek(f, s, SEEK\_SET) != -1, "fseek");

for(int i = 0; i < e - s; i++)

{

int ch = fgetc(f);

int k;

int len = strlen(W);

int sameChar;

int newCh;

int all[len];

if(ch == (int)W[0])

{

k = 0;

sameChar = 0;

for(int j = 1; j < len; j++)

{

newCh = fgetc(f);

all[k] = newCh;

k++;

if(newCh == (int)W[j])

sameChar++;

else

break;

}

if(sameChar == (len - 1))

count++;

while(k != 0)

{

k--;

check(ungetc(all[k], f) != EOF, "ungetc");

}

}

}

atomic\_fetch\_add(&globalSum, count);

fclose(f);

return NULL;

}

3. Napisati program koji kao argument komadne linije prima putanju do fajla i jednu od niski: ‘~~w’~~ , '-e' ili ‘-l’. Program pokrece dete proces koje treba da izvrsi komandu terminala wc za prosledjene argumente (npr. wc-car Ltxt za argumente dir/1.txt' i'-c'), Preusmeriti standardni izlaz iz dete process, procitati ga u roditeljskom procesu i ispisati na standardni izlaz. Ukoliko dete proces ne zavrsi uspesno, tj. exit code 0, ispisati "Neuspeh iz roditelja.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <ctype.h>

#include <sys/wait.h>

#define check(expr, msg) \

do { \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define RD\_END (0)

#define WR\_END (1)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc == 3, "argc");

char \*path = argv[1];

char \*arg = argv[2];

int cld2par[2];

check(pipe(cld2par) != -1, "pipe");

pid\_t childPid = fork();

check(childPid != -1, "fork");

if(childPid == 0)

{

close(cld2par[RD\_END]);

check(dup2(cld2par[WR\_END], STDOUT\_FILENO) != -1, "dup2");

check(execlp("wc", "wc", path, arg, NULL) != -1, "execlp");

close(cld2par[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

if(WIFEXITED(status) && (WEXITSTATUS(status) == EXIT\_SUCCESS))

{

close(cld2par[WR\_END]);

FILE \*f = fdopen(cld2par[RD\_END], "r");

check(f != NULL, "fdopen");

char \*line = NULL;

size\_t lineLen = 0;

while(getline(&line, &lineLen, f) != -1)

{

for(int i = 0; line[i] != '\0'; i++)

{

if(isspace(line[i]))

break;

printf("%c", line[i]);

}

}

fclose(f);

close(cld2par[RD\_END]);

}

else

{

printf("Neuspeh");

}

return 0;

}

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 poll interfejsa. Ispisati naziv FIFO fajla iz kog je procitano najvise karaktera (okrenite stranu).

#include <stdio.h>

#include <stdlib.h>

#include <poll.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#define check(expr, msg) \

do { \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_SIZE (1024)

int main(int argc, char \*\*argv)

{

check(argc > 1, "argc");

int nFds = argc - 1;

struct pollfd \*fds = malloc(nFds \* sizeof(struct pollfd));

check(fds != NULL, "malloc");

int \*count = calloc(nFds, sizeof(int));

check(count != NULL, "calloc");

for(int i = 0; i < nFds; i++)

{

int fd = open(argv[i+1], O\_RDONLY | O\_NONBLOCK);

check(fd != -1, "open");

count[i] = 0;

fds[i].fd = fd;

fds[i].events = POLLIN | POLLHUP | POLLERR;

fds[i].revents = 0;

}

int activeFds = nFds;

char buffer[MAX\_SIZE];

int globalMax = 0;

char \*globalName;

while(activeFds)

{

int retVal = poll(fds, nFds, -1);

check(retVal != -1, "poll");

for(int i = 0; i < nFds; i++)

{

if(fds[i].revents & POLLIN)

{

int readBytes = 0;

while((readBytes = read(fds[i].fd, buffer, MAX\_SIZE)) > 0)

{

count[i] += readBytes;

}

if(readBytes == -1)

if(errno != EAGAIN)

check(0, "read");

buffer[totalBytes] = '\0';

int localMax = totalBytes;

if(localMax > globalMax)

{

globalMax = localMax;

globalName = argv[i+1];

}

fds[i].revents = 0;

}

else if(fds[i].revents & (POLLERR | POLLHUP))

{

close(fds[i].fd);

fds[i].fd = -1;

fds[i].events = 0;

fds[i].revents = 0;

activeFds--;

}

}

}

char \*line = strrchr(globalName, '/');

char \*res = line != NULL ? line + 1 : globalName;

printf("%s\n", res);

for(int i = 0; i < nFds; i++)

if(fds[i].fd != -1)

close(fds[i].fd);

free(fds);

free(count);

return 0;

}

5. Napisati program koji kao argumente komandne linije prima nazvi objekta deljene memorije. Potrebno je ucitati strukturu [ARRAY\_MAX je 1024);

typedef struct{

sem\_t inDataReady;

sem\_t dataProcessed;

char str[ARRAY\_MAX];

}osInputData;

I prebciti sva mala slova niske str u velika I sva velika slova niske str u mala (invertovati case). Dodatno, pre bilo kakve obrade potrebno je sacekati semafor inDataReady, nakon obrade postaviti semafor dataProcessed (pretpostaviti da je ispravno inicijalizovan). NAPOMENA: Linkovati sa -lrt.

#include <stdio.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <ctype.h>

#include <semaphore.h>

#include <unistd.h>

#define check(expr, msg) \

do { \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define ARRAY\_MAX (256)

typedef struct{

sem\_t inDataReady;

sem\_t dataProcessed;

char str[ARRAY\_MAX];

}osInputData;

void \*getMemoryBlock(const char \*path, unsigned \*size);

int main(int argc, char \*\*argv)

{

check(argc == 2, "argc");

unsigned size = 0;

osInputData \*data = getMemoryBlock(argv[1], &size);

check(sem\_wait(&(data->inDataReady)) != -1, "sem\_wait");

char \*copy = data->str;

for(int i = 0; copy[i] != '\0'; i++)

{

if(islower(copy[i]))

copy[i] = toupper(copy[i]);

else if(isupper(copy[i]))

copy[i] = tolower(copy[i]);

}

//printf("%s", copy);

check(sem\_post(&(data->dataProcessed)) != -1, "sem\_post");

check(munmap(data, size) != -1, "munmap");

//check(shm\_unlink(argv[1]) != -1, "shm\_unlink");

return 0;

}

void \*getMemoryBlock(const char \*path, unsigned \*size)

{

int memFd = shm\_open(path, O\_RDWR, 0600);

check(memFd != -1, "shm\_open");

struct stat fInfo;

check(fstat(memFd, &fInfo) != -1, "fstat");

\*size = fInfo.st\_size;

void \*addr;

check((addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0)) != MAP\_FAILED, "mmap");

close(memFd);

return addr;

}

**Jun1 2018:**

1. Korisnik kao argument komandne linije prosledjuje jedan ceo broj koji predstavlja broj dana za koliko treba promeniti datum kada se program pokrece. Napisati program koji pomera datum za onoliko dana za koliko korisnik zeli. Program treba da ispise ime dana u nedelji koji odgovara novom datumu na srpskom jeziku malim slovima.

2. Napisati program koji pomocu niti izracunava histogram slova koja se nalaze u fajlu koji se programu prosledjuje kao argument komandne linije. Jedna nit broji pojavljivanja jednog slova. Dakle prva nit broji pojavljivanja slova 'a', druga slova 'b', treca slova 'c' itd. Niti ne razlikuju velika i mala slova (dakle pokrece se 26 niti). Sve karaktere koji nisu slova treba ignorisati. Na kraju rada, program stampa dobijeni histogram i slovo koje se najvise puta javilo u fajlu. Nije dozvoljeno naknadno utvrdjivanje maksimuma linearnom pretragom na osnovu zapamcenih brojeva pojavljivanja slova. Sinhronizaciju ostvariti pomocu mutex-a.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <errno.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <ctype.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define check\_pthread(err, msg) \

do{ \

int \_err = err; \

if(\_err > 0){ \

errno = \_err; \

check(0, msg); \

} \

}while(0)

#define NUMS (26)

typedef struct{

char \*file;

int idx;

}inData;

pthread\_mutex\_t globalLock = PTHREAD\_MUTEX\_INITIALIZER;

int letters[NUMS];

int max;

char letter;

void \*threadFunc(void \*arg);

int main(int argc, char\*\* argv)

{

check(argc == 2, "argc");

pthread\_t \*tids = malloc(NUMS \* sizeof(pthread\_t));

check(tids != NULL, "malloc");

inData \*args = malloc(NUMS \* sizeof(inData));

check(args != NULL, "malloc");

for(int i = 0; i < NUMS; i++)

{

args[i].file = argv[1];

args[i].idx = i;

check\_pthread(pthread\_create(&tids[i], NULL, threadFunc, &args[i]), "pthread\_create");

}

for(int i = 0; i < NUMS; i++)

{

check\_pthread(pthread\_join(tids[i], NULL), "pthread\_join");

}

for(int i = 0; i < NUMS; i++)

{

printf("%d ", letters[i]);

}

printf("\n%c\n", letter);

free(tids);

free(args);

return 0;

}

void \*threadFunc(void \*arg)

{

inData \*data = (inData\*)arg;

int count = 0;

int id = data->idx;

char \*file = data->file;

char savedLetter;

FILE \*f = fopen(file, "r");

check(f != NULL, "fopen");

char \*line = NULL;

size\_t lineLen = 0;

while(getline(&line, &lineLen, f) > 0)

{

int len = strlen(line);

for(int i = 0; i < len; i++)

{

char ch = (unsigned char)line[i];

if(!isalpha(ch))

continue;

int c = (int)(tolower(ch) - 'a');

if(c == id)

{

count++;

savedLetter = ch;

}

}

}

check(pthread\_mutex\_lock(&(globalLock)) != -1, "pthread\_mutex\_lock");

letters[id] = count;

if(count > max)

{

max = count;

letter = savedLetter;

}

check(pthread\_mutex\_unlock(&(globalLock)) != -1, "pthread\_mutex\_unlock");

fclose(f);

return NULL;

}

3. Napisati program koji u dete procesu pokrece komandu Is -1. Roditeljski proces ceka na izvrsenje komande i prikazuje nazive samo onih fajlova kod kojih ostali korisnici na sistemu imaju rwx privilegije. U slucaju da je izvrsavanje komande neuspesno, roditeljski proces ispisuje poruku neuspeh i prekida svoje izvrsavanje sa exit kodom koji je vratila komanda ls.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define RD\_END (0)

#define WR\_END (1)

#define MAX\_SIZE (256)

int main(int argc, char\*\* argv)

{

check(argc == 1, "argc");

int pipeFd[2];

check(pipe(pipeFd) != -1, "pipe");

pid\_t childPid = fork();

check(childPid != -1, "fork");

if(childPid == 0) //child

{

close(pipeFd[RD\_END]);

check(dup2(pipeFd[WR\_END], STDOUT\_FILENO) != -1, "dup2");

check(execlp("ls", "ls", "-l", NULL) != -1, "execlp");

close(pipeFd[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

if(WIFEXITED(status) && (WEXITSTATUS(status) == EXIT\_SUCCESS))

{

close(pipeFd[WR\_END]);

char \*line = NULL;

size\_t lineLen = 0;

FILE \*f = fdopen(pipeFd[RD\_END], "r");

while(getline(&line, &lineLen, f) != -1)

{

if(line[7] == 'r' && line[8] == 'w' && line[9] == 'x')

{

char \*space = strrchr(line, ' ');

printf("%s", space + 1);

}

}

free(line);

fclose(f);

close(pipeFd[RD\_END]);

}

else

{

printf("neuspeh");

exit(WEXITSTATUS(status));

}

return 0;

}

4. Program kao argument komandne linije dobija putanju do regularnog fajla. U tom fajlu, svaku otkljucanu rec program treba da zameni sa tom istom reci koja je obrnuta kao u ogledalu. U slucaju da je rec vec zakljucana, program treba da je preskoci I pokusa da izmeni sledecu rec. Na kraju rada, program ispisuje broj uspesno izmenjenih reci I broj reci koje nije uspeo da izmeni.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <errno.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_SIZE (256)

int yes = 0;

int no = 0;

int main(int argc, char\*\* argv)

{

check(argc == 2, "argc");

FILE \*f = fopen(argv[1], "r+");

check(f != NULL, "fopen");

int fd = fileno(f);

check(fd != -1, "fileno");

struct flock lock;

char read[MAX\_SIZE];

memset(read, 0, MAX\_SIZE);

while(fscanf(f, "%s", read) != EOF)

{

int len = strlen(read);

int end = len-1;

char copy[MAX\_SIZE];

for(int i = 0; i < len; i++)

{

copy[i] = read[end];

end--;

}

copy[len] = '\0';

lock.l\_type = F\_WRLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = ftell(f);

lock.l\_len = -len;

int res = fcntl(fd, F\_SETLK, &lock);

if(res == -1)

{

no++;

if(errno != EACCES && errno != EAGAIN)

check(0, "fcntl");

}

else

{

check(fseek(f, -len, SEEK\_CUR) != -1, "fseek");

fprintf(f, "%s", copy);

yes++;

}

}

lock.l\_type = F\_UNLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = 0;

lock.l\_len = 0;

check(fcntl(fd, F\_SETLK, &lock) != -1, "fcntl");

printf("%d %d", yes, no);

fclose(f);

return 0;

}

5. **Zadatak 5:** VAZNO: ZADATAK U PRIPREMI - potrebna je izmena test primera!

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

typedef struct {

sem\_t ready;

sem\_t done;

double points[MAX\_SIZE];

unsigned nPoints; /\* broj tacaka \*/

};

Program ceka na semaforu ready da podaci budu spremni za obradu, pronalazi dve najblize tacke u nizu tacaka points i na kraju postavlja semafor done. Niz points se sastoji od alternirajucih x i y koordinata tacaka (x0, y0, x1, y1 …). Na kraju rada, program stampa najkrace rastojanje koje je izracunao. Promenljiva MAX\_SIZE je 1024 (dakle najvise 512 tacaka).

*NAPOMENA: Linkovati sa* **-lrt** *i* **-lpthread***.*

*#define \_XOPEN\_SOURCE 700*

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <semaphore.h>*

*#include <sys/mman.h>*

*#include <sys/stat.h>*

*#include <fcntl.h>*

*#include <unistd.h>*

*#include <sys/types.h>*

*#include <math.h>*

*#define check(expr, msg) \*

*do{ \*

*if(!(expr)){ \*

*perror(msg); \*

*exit(EXIT\_FAILURE); \*

*} \*

*}while(0)*

*#define MAX\_SIZE (1024)*

*typedef struct{*

*sem\_t ready;*

*sem\_t done;*

*double points[MAX\_SIZE];*

*unsigned nPoints;*

*}osData;*

*void \*getMemBlock(const char \*path, unsigned \*size);*

*int main(int argc, char\*\* argv)*

*{*

*check(argc == 2, "argc");*

*unsigned size = 0;*

*osData \*data = getMemBlock(argv[1], &size);*

*check(sem\_wait(&(data->ready)) != -1, "sem\_wait");*

*int n = data->nPoints;*

*double \*points = data->points;*

*double x0, y0, x1, y1;*

*double dist;*

*double minDist;*

*int init = 0;*

*int i, j;*

*for(i = 0; i < n; i+=2)*

*{*

*x0 = points[i];*

*y0 = points[i+1];*

*for(j = 0; j < n; j+=2)*

*{*

*if(i == j)*

*continue;*

*x1 = points[j];*

*y1 = points[j+1];*

*dist = sqrt(pow(x1 - x0, 2) + pow(y1 - y0, 2));*

*if(!init)*

*{*

*minDist = dist;*

*init = 1;*

*}*

*else*

*{*

*if(dist < minDist)*

*minDist = dist;*

*}*

*}*

*}*

*check(sem\_post(&(data->done)) != -1, "sem\_post");*

*check(munmap(data, size) != -1, "munmap");*

*printf("%lf\n", minDist);*

*return 0;*

*}*

*void \*getMemBlock(const char \*path, unsigned \*size)*

*{*

*int memFd = shm\_open(path, O\_RDWR, 0600);*

*check(memFd != -1, "shm\_open");*

*struct stat fInfo;*

*check(fstat(memFd, &fInfo) != -1, "fstat");*

*\*size = fInfo.st\_size;*

*void \*addr;*

*check((addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0)) != MAP\_FAILED, "mmap");*

*close(memFd);*

*return addr;*

*}*

**Septembar1 2018:**

1. Napisati program koji kao argumente komandne linije prima 2 arguments, broj sekundi od Epobe i broj minuta za koliko treba pomeriti prosledjeno vreme u sekundama. Vreme ispisati na standardni izlaz u formatu DD/MM/YYYY HH:MM.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_SIZE (256)

#define SECONDS (60)

int main(int argc, char \*\*argv)

{

check(argc == 3, "argc");

long seconds = atol(argv[1]);

int minutes = atoi(argv[2]);

seconds += (long)(minutes \* SECONDS);

struct tm \*time = localtime(&seconds);

check(time != NULL, "localtime");

char str[MAX\_SIZE];

check(strftime(str, MAX\_SIZE, "%d/%m/%Y %R", time) != 0, "strftime");

printf("%s\n", str);

return 0;

}

2. Napisati program koji ume da pokrene i izvrsi bilo koji drugi program. Kroz argumente komandne linije program dobija putanju do drugog programa, ili njegov naziv, i listu argumenata koju je potrebno proslediti programu koji treba pokrenuti. Nakon izvrsavanja zeljenog programa, vas program ispisuje broj linija na izlazu pokrenutog programa ili „Neuspeh" u slucaju neuspeha (do neuspeha dolazi kada pokrenuti program ne zavrsi sa exit kodom 0).

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define RD\_END (0)

#define WR\_END (1)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc > 1, "argc");

int pipeFd[2];

check(pipe(pipeFd) != -1, "pipe");

pid\_t childPid = fork();

check(childPid != -1, "fork");

if(childPid == 0)

{

close(pipeFd[RD\_END]);

check(dup2(pipeFd[WR\_END], STDOUT\_FILENO) != -1, "dup2");

int n = argc;

char \*\*args = malloc(n \* sizeof(char\*));

check(args != NULL, "malloc");

int i;

for(i = 1; i < n; i++)

{

args[i-1] = malloc(strlen(argv[i] + 1));

check(args[i-1] != NULL, "malloc");

strcpy(args[i-1], argv[i]);

}

args[i-1] = NULL;

check(execvp(argv[1], args) != -1, "execvp");

close(pipeFd[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

if(WIFEXITED(status) && (WEXITSTATUS(status) == EXIT\_SUCCESS))

{

close(pipeFd[WR\_END]);

char \*line = NULL;

size\_t lineLen = 0;

FILE \*f = fdopen(pipeFd[RD\_END], "r");

check(f != NULL, "fdopen");

int num = 0;

while(getline(&line, &lineLen, f) != -1)

{

num++;

}

printf("%d\n", num);

close(pipeFd[RD\_END]);

}

else

{

printf("Neuspeh");

}

return 0;

}

3. Napisati program koji uz pomoc niti izracunava p-normu matrice. Matrica realnih brojeva (koristiti double) se ucitava sa standardnog ulaza. Prvo realan broj p, pa ceo broj vrsta m, zatim ceo broj kolona ni u narednih m redova po n brojeva koji predstavljaju elemente matrice. Norma se izracunava po sledecoj formuli:

Svaka nit Izracunava zbir stepenovanih elemenata jedne vrste (kao i stepenovanje elemenate te vrste). Niti sinhronizovati pomocu muteksa (stititi globalni zbir svih vrsta). U main funkciji nakon obrade zavrsetka svih niti ispisati p-normu (stepenovati globalni zbir na 1/p i ispisati ga).

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <math.h>

#include <errno.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define check\_pthread(err, msg) \

do{ \

int \_err = err; \

if(\_err > 0){ \

errno = \_err; \

check(0, msg); \

} \

}while(0)

typedef struct{

int idx;

}inData;

double p;

int n, m;

double \*\*matrix;

double globalSum = 0;

pthread\_mutex\_t globalLock = PTHREAD\_MUTEX\_INITIALIZER;

void \*threadFunc(void \*arg);

int main(int argc, char \*\*argv)

{

check(argc == 1, "argc");

scanf("%lf %d %d", &p, &n, &m);

matrix = malloc(n \* sizeof(double\*));

check(matrix != NULL, "malloc");

int i, j;

for(i = 0; i < n; i++)

{

matrix[i] = malloc(m \* sizeof(double));

check(matrix[i] != NULL, "malloc");

}

for(i = 0; i < n; i++)

for(j = 0; j < m; j++)

scanf("%lf", &matrix[i][j]);

pthread\_t \*tids = malloc(n \* sizeof(pthread\_t));

check(tids != NULL, "malloc");

inData \*args = malloc(n \* sizeof(inData));

check(args != NULL, "malloc");

for(i = 0; i < n; i++)

{

args[i].idx = i;

check\_pthread(pthread\_create(&tids[i], NULL, threadFunc, &args[i]), "pthread\_create");

}

for(i = 0; i < n; i++)

{

check\_pthread(pthread\_join(tids[i], NULL), "pthread\_join");

}

printf("%lf\n", pow(globalSum, 1/p));

return 0;

}

void \*threadFunc(void \*arg)

{

inData \*data = (inData\*)arg;

int k = data->idx;

double curr;

double sum = 0;

for(int i = 0; i < m; i++)

{

curr = fabs(matrix[k][i]);

sum += pow(curr, p);

}

check(pthread\_mutex\_lock(&globalLock) != -1, "pthread\_mutex\_lock");

globalSum += sum;

check(pthread\_mutex\_unlock(&globalLock) != -1, "pthread\_mutex\_unlock");

return NULL;

}

4. Napisati program koji koristi poll() funkciju da istovremeno motri vise FIFO fajlova cije se putanje prosledjuju kao argumenti komandne linije. Program iz FIFO fajlova cita sadrzaj sve dok strana koja pise ne zatvori konekciju (cita dok ima podataka). Ispisati naziv (ne putanju, nego bas naziv) FIFO fajlaiz kog je najvise puta procitan karakter ‘a’ kao I broj puta koliko je taj karakter procitan (dakle, izlaz je ime\_fajla broj\_pojavljivanja).

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <poll.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_SIZE (128)

int main(int argc, char \*\*argv)

{

check(argc > 1, "argc");

int numFifos = argc - 1;

struct pollfd \*fifos = calloc(numFifos, sizeof(struct pollfd));

check(fifos != NULL, "malloc");

FILE \*\*streams = malloc(numFifos \* sizeof(FILE\*));

check(streams != NULL, "malloc");

for(int i = 0; i < numFifos; i++)

{

int fd = open(argv[i+1], O\_RDONLY | O\_NONBLOCK);

check(fd != -1, "open");

fifos[i].fd = fd;

fifos[i].events = POLLIN | POLLERR | POLLHUP;

fifos[i].revents = 0;

streams[i] = fdopen(fd, "r");

check(streams[i] != NULL, "fdopen");

}

char name[MAX\_SIZE];

int max = 0;

int activeFifos = numFifos;

while(activeFifos)

{

int retVal = poll(fifos, numFifos, -1);

check(retVal != -1, "poll");

for(int i = 0; i < numFifos; i++)

{

if(fifos[i].revents & POLLIN)

{

int num = 0;

char \*line = NULL;

size\_t lineLen = 0;

while(getline(&line, &lineLen, streams[i]) != -1)

{

int len = (int)lineLen;

for(int i = 0; i < len; i++)

{

if(line[i] == 'a')

num++;

}

}

if(num > max)

{

max = num;

strcpy(name, argv[i+1]);

}

fifos[i].revents = 0;

}

else if(fifos[i].revents & (POLLHUP | POLLERR))

{

fclose(streams[i]);

close(fifos[i].fd);

fifos[i].fd = -1;

fifos[i].events = 0;

fifos[i].revents = 0;

activeFifos--;

}

}

}

char \*line = strrchr(name, '/');

char \*res = line != NULL ? line + 1 : name;

printf("%s %d", res, max);

free(fifos);

free(streams);

return 0;

}

5. Napisati program koji kao argumente komandne linije prima ime objekta deljene memorije. Potrebno je ucitati strukturu (ARRAY\_MAX je 1024):

typedef struct {

sem\_t dataProcessingFinished;

int array[ARRAY\_MAX];

unisgned arrayLen;

} osInputData;

Nakon ucitavanja strukture u adresni prostot programa, program ceka na signal I ukoliko mu je poslat signal SIGUSR1 menja znam svim brojevima u nizu array, dok ako je u pitanju SIGUSR2 duplira vrednosti svih elemenata niza array (mnozi ih sa 2). Nakon promene vrednosti niza array potrebno je uvecati semafor dataProcessingFinished (smatrati da je semafor dobro inicijalizovan). Ne raditi shm\_unlink.

#include <stdio.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/types.h>

#include <unistd.h>

#include <semaphore.h>

#include <signal.h>

#include <stdint.h>

#include <math.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define ARRAY\_MAX (1024)

typedef struct{

sem\_t dataProcessingFinished;

int array[ARRAY\_MAX];

unsigned arrayLen;

} OsInputData;

int sigUsr1 = 0;

int sigUsr2 = 0;

void \*getMemBlock(const char \*path, unsigned \*size);

void sigHandler(int signum);

int main(int argc, char \*\*argv)

{

check(argc == 2, "argc");

unsigned size = 0;

OsInputData \*data = getMemBlock(argv[1], &size);

printf("%jd\n", (intmax\_t)getpid());

check(signal(SIGUSR1, sigHandler) != SIG\_ERR, "signal");

check(signal(SIGUSR2, sigHandler) != SIG\_ERR, "signal");

pause();

int \*copy = data->array;

int n = data->arrayLen;

if(sigUsr1)

{

for(int i = 0; i < n; i++)

{

if(copy[i] > 0)

copy[i] = copy[i] - 2 \* copy[i];

else

copy[i] = abs(copy[i]);

}

}

else if(sigUsr2)

{

for(int i = 0; i < n; i++)

{

copy[i] = 2 \* copy[i];

}

}

for(int i = 0; i < n; i++)

{

printf("%d ", copy[i]);

}

check(sem\_post(&(data->dataProcessingFinished)) != -1, "sem\_post");

check(munmap(data, size) != -1, "munmap");

return 0;

}

void \*getMemBlock(const char \*path, unsigned \*size)

{

int memFd = shm\_open(path, O\_RDWR, 0600);

check(memFd != -1, "shm\_open");

struct stat fInfo;

check(fstat(memFd, &fInfo) != -1, "fstat");

\*size = fInfo.st\_size;

void \*addr;

check((addr = mmap(0, \*size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, memFd, 0)) != MAP\_FAILED, "mmap");

close(memFd);

return addr;

}

void sigHandler(int signum)

{

switch(signum){

case SIGUSR1:

sigUsr1 = 1;

break;

case SIGUSR2:

sigUsr2 = 1;

break;

default:

check(0, "sig");

}

}

**Januar2 2020:**

1. Napisati program koji kao argument dobija trenutak u vremenu u vidu broja sekundi od Epohe. Za prosledjeni trenutak, odnosno za godinu g prosledjenog trenutka, ispisati najblizu prestupnu godinu p takvu da je p <= g. Godina je sigurno prestupna ako je deljiva sa 4 i nije deljiva sa 100. Ukoliko je deljiva sa 100 prestupna je ako je deljiva sa 400.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

int main(int argc, char \*\*argv)

{

check(argc == 2, "argc");

long seconds = atol(argv[1]);

struct tm \*structTime = localtime(&seconds);

check(localtime != NULL, "localtime");

int year = structTime->tm\_year;

for(int i = year; i > 0; i--)

{

if((i % 4 == 0 && i % 100 != 0) || (i % 100 == 0 && i % 400 == 0))

{

printf("%d\n", i + 1900);

break;

}

}

return 0;

}

2. Napisati program koji u dete procesu pokrece komandu find. -type fi prosledjuje rezultat roditelju. Roditeljski proces treba da ispise broj fajlova cija ekstenzija odgovara ekstenziji koja se dobija kao argument komandne linije (dakle na standardni izlaz ispisati 1 broj). Pokretanje: 2 bak. Zavrsiti sa exit kodom 1 u slucaju da broj argumenata komandne linije nije dobar.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <string.h>

#include <sys/wait.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define RD\_END (0)

#define WR\_END (1)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc == 2, "argc");

char \*find = argv[1];

int len = strlen(find);

int pipeFd[2];

check(pipe(pipeFd) != -1, "pipe");

pid\_t childPid = fork();

check(childPid != -1, "fork");

if(childPid > 0) // parent

{

close(pipeFd[WR\_END]);

FILE \*f = fdopen(pipeFd[RD\_END], "r");

check(f != NULL, "fdopen");

char \*line = NULL;

size\_t lineLen = 0;

char \*dot;

int count = 0;

while(getline(&line, &lineLen, f) != -1)

{

dot = strrchr(line, '.');

if(!strncmp(dot, find, len))

count++;

}

printf("%d\n", count);

close(pipeFd[RD\_END]);

}

else

{

close(pipeFd[RD\_END]);

check(dup2(pipeFd[WR\_END], STDOUT\_FILENO) != -1, "dup2");

check(execlp("find", "find", ".", "-type", "f", NULL) != -1, "execlp");

close(pipeFd[WR\_END]);

exit(EXIT\_SUCCESS);

}

int status = 0;

check(wait(&status) != -1, "wait");

if(WIFEXITED(status) && (WEXITSTATUS(status) == EXIT\_SUCCESS))

{

}

else

{

exit(WEXITSTATUS(status));

}

return 0;

}

3. Napisati program koji kao argument komandne linije prima putanju do fajla i dve reci. U navedenom fajlu program treba da zameni sva pojavljivanja prve reci sa drugom reci (pretpostaviti da su reci iste duzine). Pre nego sto prepise staru rec, program treba da je zakljuca. Ukoliko je druge proces vec zakljucao neko pojavljivanje reci, nastaviti normalno sa obradom bez zamene reci. Dodatno, potrebno je ispisati broj reci koje nisu zamenjene (zbog toga sto je drugi proces drzao katanac). Maksimalna duzina reci u failu je 256.

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <errno.h>

#include <string.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_SIZE (256)

int main(int argc, char \*\*argv)

{

check(argc == 4, "argc");

char \*path = argv[1];

char \*find = argv[2];

char \*replace = argv[3];

int len = strlen(find);

FILE \*f = fopen(path, "r+");

check(f != NULL, "fopen");

int fd = fileno(f);

check(fd != -1, "fileno");

struct flock lock;

char str[MAX\_SIZE];

memset(str, 0, MAX\_SIZE);

int count = 0;

while(fscanf(f, "%s", str) != EOF)

{

if(strcmp(str, find))

continue;

lock.l\_type = F\_WRLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = ftell(f);

lock.l\_len = -len;

int res = fcntl(fd, F\_SETLK, &lock);

if(res == -1)

{

if(errno != EACCES && errno != EAGAIN)

check(0, "fcntl");

else

count++;

}

else

{

check(lseek(fd, -len, SEEK\_CUR) != (off\_t) - 1, "lseek");

fprintf(f, "%s", replace);

}

}

lock.l\_type = F\_UNLCK;

lock.l\_whence = SEEK\_SET;

lock.l\_start = 0;

lock.l\_len = 0;

check(fcntl(fd, F\_SETLK, &lock) != -1, "fcntl");

printf("%d\n", count);

fclose(f);

return 0;

}

4. Napisati program koji kao argumente komandne linije prima putanje do FIFO fajlova. Program istovremeno za sve fajlove ceka na dogadjaj da je kroz neki FIFO stigao podatak koriscenjem epoll interfejsa. Kroz FIFO fajlove pristizu realni brojevi, cita se sve dok se ne zatvore svi FIFO fajlovi na strani koja pise u FIFO. Na standardni izlaz ispisati maksimalni procitan realni broj i naziv FIFO fajla (ne putanju nego NAZIV).

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <sys/epoll.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <unistd.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_EVENTS (8)

#define TIME\_OUT (-1)

int main(int argc, char \*\*argv)

{

check(argc > 1, "argc");

int numFifos = argc - 1;

int epollFd = epoll\_create(numFifos);

check(epollFd != -1, "epoll\_create");

int \*fds = malloc(numFifos \* sizeof(int));

check(fds != NULL, "malloc");

double \*maxs = malloc(numFifos \* sizeof(double));

check(maxs != NULL, "malloc");

struct epoll\_event currentEvent;

for(int i = 0; i < numFifos; i++)

{

memset(&currentEvent, 0, sizeof(struct epoll\_event));

currentEvent.events = EPOLLIN;

currentEvent.data.fd = open(argv[i+1], O\_RDONLY | O\_NONBLOCK);

check(currentEvent.data.fd != -1, "open");

fds[i] = currentEvent.data.fd;

check(epoll\_ctl(epollFd, EPOLL\_CTL\_ADD, currentEvent.data.fd, &currentEvent) != -1, "epoll\_ctl");

}

int activeFifos = numFifos;

struct epoll\_event readyFds[MAX\_EVENTS];

while(activeFifos)

{

int retVal = epoll\_wait(epollFd, readyFds, MAX\_EVENTS, TIME\_OUT);

check(retVal != -1, "epoll\_wait");

for(int i = 0; i < retVal; i++)

{

if(readyFds[i].events & EPOLLIN)

{

double num;

double currMax;

int init = 0;

FILE \*f = fdopen(readyFds[i].data.fd, "r");

check(f != NULL, "fdopen");

while(fscanf(f, "%lf ", &num) == 1)

{

if(!init)

{

currMax = num;

init = 1;

}

else

{

if(num > currMax)

currMax = num;

}

}

for(int j = 0; j < numFifos; j++)

{

if(fds[j] == readyFds[i].data.fd)

maxs[j] = currMax;

}

}

else if(readyFds[i].events & (EPOLLHUP | EPOLLERR))

{

close(readyFds[i].data.fd);

activeFifos--;

}

}

}

int maxFd = 0;

double max = maxs[0];

for(int i = 0; i < numFifos; i++)

{

if(maxs[i] > max)

{

max = maxs[i];

maxFd = i;

}

}

char \*line = strrchr(argv[maxFd + 1], '/');

char \*maxName = line != NULL ? line + 1 : argv[maxFd + 1];

printf("%g %s\n", max, maxName);

free(fds);

free(maxs);

close(epollFd);

return 0;

}

4\_poll.c

#define \_XOPEN\_SOURCE 700

#include <stdio.h>

#include <stdlib.h>

#include <sys/poll.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <unistd.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

#define MAX\_EVENTS (8)

#define TIME\_OUT (-1)

int main(int argc, char \*\*argv)

{

check(argc > 1, "argc");

int numFifos = argc - 1;

struct pollfd \*fds = malloc(numFifos \* sizeof(struct pollfd));

check(fds != NULL, "malloc");

FILE \*\*streams = malloc(numFifos \* sizeof(FILE\*));

check(streams != NULL, "malloc");

for(int i = 0; i < numFifos; i++)

{

int fd = open(argv[i+1], O\_RDONLY | O\_NONBLOCK);

check(fd != -1, "open");

streams[i] = fdopen(fd, "r");

check(streams[i] != NULL, "fdopen");

fds[i].fd = fd;

fds[i].events = POLLIN;

fds[i].revents = 0;

}

int activeFifos = numFifos;

char \*maxName;

double max;

int init = 0;

while(activeFifos)

{

int retVal = poll(fds, numFifos, -1);

check(retVal != -1, "poll");

for(int i = 0; i < numFifos; i++)

{

if(fds[i].revents & POLLIN)

{

double num;

while(fscanf(streams[i], "%lf", &num) == 1)

{

if(!init)

{

init = 1;

maxName = argv[i+1];

max = num;

}

else

{

if(num > max)

{

max = num;

maxName = argv[i+1];

}

}

}

fds[i].revents = 0;

}

else if(fds[i].revents & (POLLHUP | POLLERR))

{

close(fds[i].fd);

fds[i].fd = -1;

fds[i].events = 0;

fds[i].revents = 0;

activeFifos--;

}

}

}

char \*line = strrchr(maxName, '/');

char \*res = line != NULL ? line + 1 : maxName;

printf("%g %s", max, res);

free(streams);

free(fds);

return 0;

}

5. Program kao argument komandne linije prima rec w koja bi trebalo da sadrzi samo velika slova (potrebno je proveriti ovo, pogledati test primere). Ispisati vrednosti svih promenljivih okruzenja takvih da pocinju sa w.

#include <stdio.h>

#include <stdlib.h>

#define check(expr, msg) \

do{ \

if(!(expr)){ \

perror(msg); \

exit(EXIT\_FAILURE); \

} \

}while(0)

int main(int argc, char \*\*argv)

{

check(argc > 1, "argc");

return 0;

}