Zadanie: zadanie_3_01 Kompilacia: gcc zadanie_3_01.c -o zadanie_3_01 -lpthread

Implementujte synchronizaciu vlakien pre ulohu cyklickej montaze vyrobku, ktory pozostava z <n> komponent. Vyroba moze zacat, len ak je k dispozicii vsetkych <n> komponent. Dodavku roznych komponent opakovane zabezpecuju vlakna "dodavatel" a "vyrobca". Napiste program typu "randevue", ktory bude synchronizovat vlakna pre opakovany prisun komponent a opakovanu montaz vyrobku pomocou semafora. Kazde vlakno sluzi na dodanie jedneho druhu komponenty. Dalsie komponenty mozu vlakna dodavat len po zacati montaze. Trvanie dodavky komponent a samotnej montaze vlaknami simulujte funkciou sleep s nahodne generovaym oneskorenim z intervalu <1,3>. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_01 [-h] <n>

Output: Poradove cislo vyrobku s informaciou o zacati a ukonceni montaze.

Zadanie: zadanie_3_02 Kompilacia: gcc zadanie_3_02.c -o zadanie_3_02 -lpthread

Napiste program, ktory bude kopirovat <subor 1> do <suboru 2> cez buffer v pamati konecnej velkosti <velkost buffra>. Jedno vlakno bude zo suboru citat a zapisovat do buffra, dalsie citat z buffra a zapisovat do noveho suboru. Na pristup k buffru pouzite zamku a semafory na testovanie podmienok, ci je buffer prazdny alebo plny. Na zistenie poctu pristupov jednotlivych vlakien k suboru pre citanie resp. zapis pouzite specificku premennu. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_02 [-h] <subor1> <subor2> <velkost buffra>

Zadanie: zadanie_3_03 Kompilacia: gcc zadanie_3_03.c -o zadanie_3_03 -lpthread

Implementujte konkurentnu verziu vypoctu velkosti vektora dlzky <k> <n> vlaknami (parametre su dane argumentami programu). Hlavne udaje su dostupne vsetkym vlaknam v globalnej strukture S_VEC a kazde z vlakien pracuje s roznymi castami tychto udajov. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

```
struct S_VEC {
        double *vec;  // vector
        double sum;  // sucet
        int k;          // k rozmer vektorov
}
```

Syntax: zadanie_3_03 [-h] <n> <k>

Output: 'velkost vektora'

Zadanie: zadanie_3_04 Kompilacia: gcc zadanie_3_04.c -o zadanie_3_04 -lpthread

Implementujte program, ktory spocita pocet vyskytov zadaneho <pismena> v <subore>. Pismeno a nazov suboru budu parametrami programu. Program bude obsahovat viacero pracovnych vlakien, ktorych pocet <pocet_vlakien> bude tiez parametrom programu. Tieto vlakna budu citat obsah suboru po blokoch velkosti <block_size> naraz a spocitavat pocet vyskytov zadaneho pismena v nacitanej casti suboru. Tento pocet pripocitaju do globalnej premennej, v ktorej sa bude udrzovat celkovy pocet vyskytov. Na zistenie poctu pristupov kazdeho vlakna k suboru pre citanie pouzite specificku premennu. Ak nie je zadany parameter <block_size>, vlakna citaju naraz cely subor. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_04 [-h] <subor> <pismeno> <pocet_vlakien> <block_size>

Output: 'pocet vyskytov pismena'

Zadanie: zadanie_3_05 Kompilacia: gcc zadanie_3_05.c -o zadanie_3_05 -lpthread
Napiste program typu konkurentny server, ktory bude komunikovat cez <schranku> (socket) s viacerymi klientami, ktorych max. pocet <pocet spojeni> bude argumentom programu. Hlavne vlakno servera bude cakat na spojenia s klientmi a pre samotnu komunikaciu s klientom vytvori pracovne vlakno. Pre komunikaciu pouzite lubovolnu textovu spravu, ktora sa po vyslani pracovnym vlaknom zobrazi na strane klienta a po prijati pracovnym vlaknom zobrazi na strane servera spolu s <tid> vlakna, ktore zabezpecuje komunikaciu. Vytvorenie spolupracujuceho klienta je v zadani cislo 20. Ukoncenie programov moze byt normalne, ale aj prerusenim (signalom SIGINT), pricom musia byt uvolnene vsetky zdroje. Asynchronny signal SIGINT musi byt osetreny synchronne specialnym cakajucim vlaknom a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_05 [-h] <adresa schranky> <pocet spojeni>

Zadanie: zadanie_3_06 Kompilacia: gcc zadanie_3_06.c -o zadanie_3_06 -lpthread
Implementujte vypocet maxima prvkov <n> rozmerneho pola <k> vlaknami. Pracovne vlakna opakovane prijimaju interval pola, ktory maju spracovat od hlavneho vlakna dovtedy, kym nebude najdene maximum prvkov celeho pola. Hlavne vlakno spocita priebezne maximum po dodani ciastkovych maxim od prac. vlakien. Synchronizaciu vlakien zabezpecte podmienkovymi premennymi. Prvky pola vytvorte generovanim nahodnych cisiel zo zvoleneho intervalu. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_06 [-h] <n> <k>
Output: 'maximum'

Zadanie: zadanie_3_07 Kompilacia: gcc zadanie_3_07.c -o zadanie_3_07 -lpthread
Implementujte ulohu obsluhy zakaznikov cakajucich vo fronte pomocou vlakien. Ak je fronta plna, zakaznik necaka (uspi sa na nahodny cas a znova sa pokusi vstupit do fronty), v opacnom pripade sa zaradi do fronty. Zakaznik, ktory je na rade, z fronty vystupi a je obsluzeny. Program overte minimalne pre <n> vlakien zakaznika a zvolenu <kapacitu fronty> (su dane v argumentoch programu) a jedno vlakno obsluhy. Trvanie obsluhy zakaznika simulujte funkciou sleep a zakaznicke vlakna nech opakovane ziadaju o obsluhu s oneskorenim generovanym nahodne. Program nech zobrazuje informacie o pocte vlakien vo fronte a informaciu, ktore vlakno je obsluhovane. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia). Uloha sa riesi ako "problem spiaceho holica". Pri rieseni ulohy pouzite zamky a podmienkove premenne.

Syntax: zadanie_3_07 [-h] <n> <kapacita fronty>
Output: '<pocet vlakien vo fronte>' '<tid obsluhovaneho vlakna (0 ak ziadne)>'

Zadanie: zadanie_3_08 Kompilacia: gcc zadanie_3_08.c -o zadanie_3_08 -lpthread
Implementujte synchronizaciu vlakien pre ulohu cyklickej montaze vyrobku, ktory pozostava z <n> komponent. Vyroba moze zacat, len ak je k dispozicii vsetkych <n> komponent. Dodavku roznych komponent opakovane zabezpecuju vlakna "dodavatel" a "vyrobca". Napiste program typu "randevue", ktory bude synchronizovat vlakna pre opakovany prisun komponent a opakovanu montaz vyrobku pomocou podmienkovej premennej. Kazde vlakno sluzi na dodanie jedneho druhu komponenty. Dalsie komponenty mozu vlakna dodavat len po zacati montaze. Trvanie dodavky komponent a samotnej montaze vlaknami simulujte funkciou sleep s nahodnym oneskorenim z intervalu <1,3>. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_08 [-h] <n>
Output: Poradove cislo vyrobku s informaciou o zacati a ukonceni montaze.

Zadanie: zadanie_3_09 Kompilacia: gcc zadanie_3_09.c -o zadanie_3_09 -lpthread
Napiste program, v ktorom vytvori hlavne vlakno tri dalsie vlakna. Dve z tychto vlakien budu inkrementovat globalnu premennu <count>, pokial tato nedosiahne hodnotu <tcount>. Nasledne hodnotu <count> tretie vlakno vynuluje a prve dve vlakna mozu pokracovat v jej inkrementovani. Na koordinaciu vlakien pouzite podmienkove premenne. Do funkcii vlakien vlozte kontrolne vypisy (<tid> vlakna, ktore dosiahlo danu hodnotu <count>, resp. novu hodnotu <count> nastavenu tretim vlaknom). Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia). Hodnota <tcount> je dana argumentom programu.

Syntax: zadanie_3_09 [-h] <tcount>
Output: 'tid vlakna' 'count'

Zadanie: zadanie_3_10 Kompilacia: gcc zadanie_3_10.c -o zadanie_3_10 -lpthread
Implementujte ulohu obsluhy zakaznikov cakajucich vo fronte pomocou vlakien. Ak je fronta plna, zakaznik necaka (uspi sa a o nahodny cas sa znova pokusi vstupit do fronty), v opacnom pripade sa zaradi do fronty. Zakaznik, ktory je na rade, z fronty vystupi a je obsluzeny. Program overte minimalne pre <n> vlakien zakaznika a zvolenu <kapacitu fronty> (su dane v argumentoch programu) a jedno vlakno obsluhy. Trvanie obsluhy zakaznika simulujte funkciou sleep a zakaznicke vlakna nech opakovane ziadaju o obsluhu s oneskorenim generovanym nahodne. Program nech zobrazuje informacie o pocte vlakien vo fronte a informaciu, ktore vlakno je obsluhovane. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia). Uloha sa riesi ako "problem spiaceho holica". Pri rieseni ulohy pouzite zamok a semafory.

Syntax: zadanie_3_10 [-h] <n> <kapacita fronty>
Output: '<pocet vlakien vo fronte>' '<tid obsluhovaneho vlakna (0 ak ziadne)>'

Zadanie: zadanie_3_11 Kompilacia: gcc zadanie_3_11.c -o zadanie_3_11 -lpthread -lrt
Implementujte program, v ktorom budu vlakna vyberat ulohy z fronty a vykonavat ich. Ulohy bude do fronty vkladat ine vlakno v pravidelnych intervaloch dlzky <k> (pouzite casovac), kde <k> bude argumentom programu. Vlakno vlozi ulohu do fronty len vtedy, ak nie je plna (kapacita fronty <m> je dana argumentom programu). Na synchronizaciu vkladania pouzite semafor. Hlavne vlakno vytvori <n> (dane argumentom programu) pracovnych vlakien, ktore budu blokovane semaforom, ak je fronta prazdna. V opacnom pripade budu ulohy z fronty vyberat a vykonavat ich. Dlzku vykonu ulohy simulujte funkciou sleep s nahodnym argumentom z intervalu <1,10>. Program nech zobrazuje informacie o pocte uloh vo fronte a informaciu, ktore vlakna spracuvaju ktore ulohy. Navrhnite vhodny sposob, ako a kedy dane informacie zobrazit. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_11 [-h] <n> <k> <m>
Output: '<pocet uloh vo fronte>' '<tid vlakna a identifikator ulohy>'

Zadanie: zadanie_3_13 Kompilacia: gcc zadanie_3_13.c -o zadanie_3_13 -lpthread
Napiste program, ktory bude kopirovat <subor 1> do <suboru 2> cez buffer v pamati konecnej velkosti. Jedno vlakno bude zo suboru citat a zapisovat do buffra, dalsie citat z buffra a zapisovat do noveho suboru. Na pristup k buffru pouzite zamku a podmienkove premenne na testovanie podmienok, ci je buffer prazdny alebo plny. Na zistenie poctu pristupov jednotlivych vlakien k suboru pre citanie resp. zapis pouzite specificku premennu. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_13 [-h] <velkost buffra> <subor 1> <subor 2>

Zadanie: zadanie_3_12 Kompilacia: gcc zadanie_3_12.c -o zadanie_3_12 -lpthread -lrt
Implementujte program, v ktorom budu vlakna vyberat ulohy z fronty uloh a vykonavat ich. Ulohy bude do fronty vkladat ine vlakno v pravidelnych intervaloch dlzky <k> (pouzite casovac), <k> je argumentom programu. Vlakno ulohu vlozi len v pripade, ze fronta nie je plna (kapacita fronty <m> je dana argumentom programu). Na synchronizaciu vkladania pouzite podmienkovu premennu. Hlavne vlakno vytvori <n> (dane argumentom programu) pracovnych vlakien. Ak je fronta prazdna, pracovne vlakno sa zablokuje na podmienkovej premennej, v opacnom pripade vyberie uohu z fronty a vykona ju. Dlzku vykonu ulohy simulujte funkciou sleep s nahodnym argumentom z intervalu <1,10>. Program nech zobrazuje informacie o pocte uloh vo fronte a informaciu, ktore vlakna spracuvaju ktore ulohy. Navrhnite vhodny sposob, ako a kedy dane informacie zobrazit. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_12 [-h] <n> <k> <m>
Output: '<pocet uloh vo fronte>' '<tid vlakna a identifikator ulohy>'

Zadanie: zadanie_3_14 Kompilacia: gcc zadanie_3_14.c -o zadanie_3_14 -lpthread
Napiste program typu readers/writers, ktory umozni sucasne zapisovat (svoje <tid> a <cas zapisu>) len jednemu vlaknu. Citat moze sucasne viac vlakien. Vypisujuce vlakno vypise precitanu informaciu spolu so svojim <tid>. Vlakna budu pracovat s konecnym buffrom o velkosti <velkost buffra> danej v argumente programu. Program implementujte pre viac zapisovatelov a viac citatelov. Zapisovacie a citacie vlakna sa vzdy po praci s buffrom uspia na nahodny cas generovany z intervalu <1,3>. Ulohu rieste pomocou zamky a podmienkovych premenych. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_14 [-h] <velkost buffra>
Output: <id vypisujuceho vlakna> <obsah buffra>

Zadanie: zadanie_3_15 Kompilacia: gcc zadanie_3_15.c -o zadanie_3_15 -lpthread
Napiste program typu readers/writers, ktory umozni sucasne zapisovat (svoje <tid> a <cas zapisu>) len jednemu vlaknu. Citat moze sucasne viac vlakien. Vypisujuce vlakno vypise precitanu informaciu spolu so svoji <tid>. Vlakna budu pracovat s konecnym buffrom o velkosti <velkost buffra> danej v argumente programu. Program implementujte pre viac zapisovatelov a viac citatelov. Zapisovacie a citacie vlakna sa vzdy po praci s buffrom uspia na nahodny cas generovany z intervalu <1,3>. Ulohu rieste pomocou zamky a semaforov. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_15 [-h] <velkost buffra>
Output: <id vypisujuceho vlakna> <obsah buffra>

Zadanie: zadanie_3_16 Kompilacia: gcc zadanie_3_16.c -o zadanie_3_16 -lpthread -lrt
Vytvorte program, v ktorom budu dve vlakna inkrementovat kazde svoje pocitadlo po zadanu hodnotu <n1> resp. <n2> dane argumentami programu. Inkrementacia musi nastat, ked pride signal casoveho prerusenia (od casovaca) nasledovne: Jedno vlakno bude inkrementovat v parnych nasobkoch casovej jednodky o hodnotu nasobku casovej jednotky <c>, druhe v neparnych tiez o hodnotu nasobku casovej jednotky. Na synchronizaciu vlakien pouzite podmienkove premenne. Vlakna nech si pamataju pocet inkrementovani vo svojej specifickej premennej. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_16 [-h] <n1> <n2> <c>
Output: 'pocet inkrementovani kazdym vlaknom'

Zadanie: zadanie_3_17 Kompilacia: gcc zadanie_3_17.c -o zadanie_3_17 -lpthread -lrt
Implementujte program, v ktorom budu 3 pracovne vlakna dekrementovat globalne <pocitadlo> z pociatocnej hodnoty <value> danej v argumente programu na 0. Vlakna budu dekrementovat <pocitadlo> len po prichode signalu casoveho prerusenia od casovaca nasledovne: Prve vlakno, ak je nasobok casovej jednotky delitelny 2, druhe delitelny 3 a tretie delitelny 4. Pocet dekrementovani nech si kazde vlakno pamata ako svoju specificku premennu. Vlakna synchronizujte podmienkovymi premennymi. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_17 [-h] <value>

Output: 'pocet dekrementovani kazdym vlaknom'

Zadanie: zadanie_3_18 Kompilacia: gcc zadanie_3_18.c -o zadanie_3_18 -lpthread
Napiste program typu vyrobca/konzument pre viac vyrobcov a/alebo konzumentov reprezentovanych vlaknami, ktore budu vkladat a vyberat z ADT zasobnik. Vkladanie a vyber nemoze byt sucasny. Na vzajomne vylucenia vlakien pouzite zamku. Pri preteceni alebo podteceni zasobnika musia vlakna cakat (pouzite semafory). <Kapacita> zasobnika je argumentom programu. Vyrobcovia budu v nekonnecnej slucke zapisovat do zasobnika nahodne cele cisla v rozsahu 1 az 10. Kazdy vyrobca po ukonceni svojho behu vypise sucet cisiel, ktore do zasobnika zapisal. Konzumenti budu v slucke cisla zo zasobnika vyberat, pokial zasobnik nebude prazdny. Ak konzument zisti, ze zasobnik je prazdny, ukonci svoj beh a vypise sucet precitanych cisiel. Za predpokladu ze je zasobnik na zaciatku a konci prazdny, suma cisiel, ktore vypisu vyrobcovia, sa musi rovnat sume, ktoru vypisu konzumenti. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia). Pritom za zobrazi akualny stav zasobnika (sucet cisiel) a sucty cisiel, ktore zapisali vsetci vyrobcovia a ktore vybrali vsetci konzumenti.

Syntax: zadanie_3_18 [-h] <kapacita>

Zadanie: zadanie_3_19 Kompilacia: gcc zadanie_3_19.c -o zadanie_3_19 -lpthread
Napiste program typu vyrobca/konzument pre viac vyrobcov a/alebo konzumentov reprezentovane vlaknami, ktore budu vkladat a vyberat z ADT zasobnik. Vkladanie a vyber nemoze byt sucasny. Na vzajomne vylucenia vlakien pouzite zamku. Pri preteceni alebo podteceni zasobnika musia vlakna cakat (pouzite podmienkove premenne). <Kapacita> zasobnika je argumentom programu. Vyrobcovia budu v nekonnecnej slucke zapisovat do zasobnika nahodne cele cisla v rozsahu 1 az 10. Kazdy vyrobca po ukonceni svojho behu vypise sucet cisiel, ktore do zasobnika zapisal. Konzumenti budu v slucke cisla zo zasobnika vyberat, pokial zasobnik nebude prazdny. Ak konzument zisti, ze zasobnik je prazdny, ukonci svoj beh a vypise sucet precitanych cisiel. Za predpokladu, ze je zasobnik na zaciatku a konci prazdny, suma cisiel, ktore vypisu vyrobcovia, sa musi rovnat sume cisiel, ktore vypisu konzumenti. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia). Pritom za zobrazi akualny stav zasobnika (sucet cisiel) a sucty cisiel, ktore zapisali vsetci vyrobcovia a ktore vybrali vsetci konzumenti.

Syntax: zadanie_3_19 [-h] <kapacita>

Zadanie: zadanie_3_20 Kompilacia: gcc zadanie_3_20.c -o zadanie_3_20 -lpthread

Napiste program typu klient, ktory budu komunikovat cez <schranku> so serverom (sockets). Program klienta bude simulovat spojenie so serverom pre viacerych klientov vytvorenim maximalne <n> vlakien (pocet pozadovanych spojeni so serverom) zabezpecujucich komunikaciu. Pre komunikaciu pouzite vstup z klavesnice, ktory sa po vyslani zobrazi na strane servera a po prijati zobrazi na strane klienta spolu s <tid> vlakna, ktore zabezpecilo komunikaciu. Vytvorenie spolupracujuceho servera je v zadani cislo 05. Ukoncenie programov moze byt normalne, ale aj prerusenim pomocou signalu SIGINT. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_20 [-h] <adresa schranky> <n>

Zadanie: zadanie_3_21 Kompilacia: gcc zadanie_3_21.c -o zadanie_3_21 -lpthread

Implementujte simulaciu spracovania instrukcii v ADT fronta kapacity <k>. Instrukcie su dvoch druhov: pre nasobenie a pre scitanie. Instrukcie su vykonavane procesorom, ktory obsahuje jednu nasobicku a jednu scitacku. Program bude obsahovat styri vlakna: pre simulaciu vyberu instrukcii z fronty, pre simulaciu vkladania instrukcii do fronty, pre simulaciu nasobenia a scitania. Typ vkladanej instrukcie generujte nahodne. Instrukcie su postupne vyberane z fronty a predavane na spracovanie nasobicke alebo scitacke. Ak je nasobicka alebo scitacka obsadena, tak instrukcia musi cakat na spracovanie. Na synchronizaciu pouzite podmienkovu premennu. Dlzku vykonu instrukcie simulujte funkciou sleep s nahodnym argumentom z intervalu <1,3>. Synchronizaciu vyberu a vkladania instrukcii do fronty zabezpecte semaformi. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia).

Syntax: zadanie_3_21 [-h] <k>

Zadanie: zadanie_3_22 Kompilacia: gcc zadanie_3_22.c -o zadanie_3_22 -lpthread

Implementujte konkurentnu verziu vypoctu nasobenia matice vektorom. Rozmer stvorcovej matice (a tym aj vektora) <n> bude dane v argumente programu. Maticu aj vektor inicializujte nahodnymi cislami vo vhodnom rozsahu. Paralelny vypocet musi zabezpecovat minimalne <n> vlakien. Navrhnite vhodny algoritmus paralelneho nasobenia a strukturu globalne dostupnych udajov jednotlivych vlakien. Program musi reagovat na stlacenie ctrl+c ukoncenim. Tento asynchronny signal musi byt osetreny synchronne specialnym vlaknom cakajucim na SIGINT a nastavenim priznaku ukoncenia. Hlavne vlakno bude na podmienkovej premenej cakat na jeho nastavenie a zrusi ostatne vlakna. Zrusene vlakna musia uvolnit vsetky zdroje (rieste bud cleanup handlerom alebo zvolenim bodov zrusenia). Vystup by mal byt vhodne riadkovo aj stlpcovo zarovnany (vid priklad na konci).

Syntax: zadanie_3_22 [-h] <n>

Output: '<vstupny vektor> x <matica> = <vystupny vektor>'

Priklad vystupu:

```
                ( 1 0 0 )
( 0 1 0 ) x ( 0 1 1 ) = ( 0 1 1 )
                ( 1 0 1 )
```