Stiahnite si súbor `vlakna.c`.

Postupne vypracujte jednotlivé úlohy. Úlohy na seba nadväzujú, preto pri vypracovávaní ďalšej úlohy využite kód z predchádzajúcej úlohy. Niektoré časti z predchádzajúcej úlohy ale nebudú potrebné. Pred vypracovaním každej ďalšej úlohy vytvorte kópiu zdrojového súboru.

Pozor pripravenú funkciu `Print` môžete použiť až od úlohy 4.

Pri kompilácii nezabudnite pridať parameter pre linker `-lpthread`.

1) Vytvorte niekoľko pracovných vlákien. Každé pracovné vlákno nech počká náhodný čas, potom vytlačí reťazec a ukončí svoju činnosť. Hlavné vlákno bude čakať na ukončenie pracovných vlákien. Potom vytlačí hlásenie o ukončení všetkých pracovných vlákien.

2) Upravte program tak, aby každé pracovné vlákno dostalo ako parameter vlastnú štruktúru typu `Data`. Štruktúra obsahuje označenie vlákna pre výpis (položka `label`) a počítadlo (položka `counter`). Štruktúry dynamicky alokujte a inicializujte volaním funkcie `DataCreate`. Nech sa v hlavnom vlákne nepamätajú pointre na vytvorené štruktúry. Nech pracovné vlákna nastavia položku `counter` na náhodnú hodnotu. Potom nech vypíšu svoje označenie (položka `label`) aj nastavenú hodnotu položky `counter`. Nech pracovné vlákna vrátia pointer na štruktúru, ktorú dostali ako parameter. Nech hlavné vlákno čaká na ukončenie pracovných vlákien. Nech pri ukončení získa pointer na vrátenú štruktúru a vypíše jej obsah a dealokuje ju. Nakoniec nech hlavne vlákno vypíše správu o ukončení všetkých pracovných vlákien.

3) Upravte program tak, aby každé pracovné vlákno vykonávalo nekonečný cyklus. Teraz môže byť návratovou hodnotou vlákna za cyklom `NULL`, pretože sa táto hodnota už nebude využívať. Nech pracovné vlákno v každom opakovaní cyklu inkrementuje položku `counter` a vypisuje svoje označenie (položka `label`) a počet opakovaní cyklu (inkrementovaná položka `counter`). Po výpise sa vždy uspí na náhodný čas. Hlavné vlákno upravte tak, aby po vytvorení pracovných vlákien počkalo určitý čas (napr. 10 sekúnd), potom zrušilo všetky pracovné vlákna (funkciou `pthread_cancel`), potom počkalo na ukončenie všetkých pracovných vlákien a vypísalo hlásenie o ukončení všetkých pracovných vlákien. Pracovné vlákna musia pri ukončení vypísať svoje označenie (položka `label`) a počet opakovaní cyklu ktoré vykonali (položka `counter`). Výpis a dealokáciu teraz nebude vykonávať hlavné vlákno, ale cleanup handler. Ako cleanup handler využite pripravenú funkciu `DataPrintAndDestroy`.

4) V tejto úlohe využite na výpis a dealokáciu štruktúry `Data` špecifické premenné. Zakomentujte nastavenie cleanup handlera, pretože jeho funkciu nahradí „deštruktor“ priradený ku kľúču špecifickej premennej. Na začiatku programu je definovaná globálna premenná `dataKey` reprezentujúca kľúč ku špecifickým premenným vlákien. Pred vytvorením pracovných vlákien vytvorte kľúč pre špecifické premenné a ako „deštruktor“ nastavte pripravenú funkciu `DataPrintAndDestroy`. V pracovných vláknach (na začiatku funkcie) priraďte ku kľúču pointer na vstupnú štruktúru `Data`. Teraz už napríklad môžete používať pripravenú funkciu `Print`, ktorá na základe kľúča pristúpi ku špecifickej premennej pracovného vlákna.

5) Do pracovných vlákien pridajte volanie funkcie `pthread_testcancel` a upravte kód tak, aby (počas vykonávania slučky) bolo vlákno možné zrušiť iba pri volaní tejto funkcie.