Pokračujte na ďalších projektoch z predchádzajúceho cvičenia.

1. zadanie (podadresár `msg_queue`)

Zadanie je zamerané na komunikáciu procesov pomocou fronty správ. Výstupom budú 2 programy, jeden bude do fronty správ posielať údaje, druhý ich bude čítať. Komunikácia medzi procesmi bude prebiehať tak, že najprv sa prenesie počet študentov a potom sa budú prenášať informácie o študentoch.

a) Súbor `commonMsgQueue.h` obsahuje definíciu spoločných makier a štruktúr. Zmeňte definíciu makra `MSG_QUEUE_KEY`, obsahujúceho kľúč fronty správ, aby ste nepoužívali tú istú frontu správ ako ostatný. Tento kľúč jedinečne identifikuje frontu správ v celom systéme.
b) Kedy je možné použiť ako kľuč identifikujúci frontu správ `IPC_PRIVATE`?
c) V súboroch `msgQueueSender.c` aj `msgQueueReceiver.c` doplňte získanie id fronty správ podľa kľúča fronty správ. V `msgQueueSender.c` nastavte parametre `msgget` tak, aby proces vytvoril novú frontu správ (ak fronta správ už existuje, tak nech sa program ukončí chybou).
d) Otestujte `msgQueueSender`. Pomocou príkazu `ipcs -q` zadaného v konzole, môžete zistiť informácie o frontach správ v systéme. Program `msgQueueSender` by mal vytvoriť frontu. Odstrániť frontu môžete príkazom `ipcrm` zadaným z konzoly.
e) V `msgQueueSender.c` doplňte poslanie počtu študentov do fronty (pošlite správu obsahujúcu číslo). Jednoduchý test môžete vykonať pomocou príkazu `ipcs -q`, ktorý umožní zistiť počet správ a množstvo údajov vo fronte.
f) Do `msgQueueReceiver.c` doplňte načítanie počtu študentov z fronty správ.
g) Do `msgQueueReceiver.c` doplňte zrušenie fronty správ v systéme. Otestujte program.
h) Do `msgQueueSender.c` doplňte posielanie informácii o študentoch.
i) Do `msgQueueReceiver.c` doplňte načítanie a výpis informácii o študentoch.
j) Ako by ste upravili program tak, aby komunikácia cez jednu frontu správ bola obojsmerná? Nápoveď: typy správ?

2. zadanie (podadresár `socket`)

Zadanie je zamerané na komunikáciu procesov pomocou soketov (schránok). Výstupom budú 2 programy (server a klient). Po vytvorení spojenia klient pošle serveru údaje o študentovi, sever príjme údaje, modifikuje ich a pošle klientovi. Klient príjme a vytlačí prijaté údaje.

Na komunikáciu bude využitý soket, ktorý bude v súborovom systéme. Čiže komunikujúce procesy musia bežať na tom istom systéme. V súbore `commonSocket.h` je definícia názvu soketu v súborovom systéme. Cesta ku soketu je relatívna, preto je potrebné, aby boli klient aj server spustené v tom istom adresári.

a) V kóde servera doplňte vytvorenie soketu v UNIX domain (funkcia `socket`).
b) V kóde servera doplňte previazanie mena (adresy v súborovom systéme) so soketom (funkcia `bind`). V `man socket` sú odkazy na manuály, ktoré sa zaoberajú jednotlivými doménami komunikácie pomocou soketov. V týchto manuáloch sú popísané aj jednotlivé verzie štruktúry `sockaddr`, ktorú budete potrebovať pri volaní funkcie `bind`.
c) V kóde servera doplňte nastavenie soketu ako pasívneho soketu a nastavenie maximálnej veľkosti fronty prichádzajúcich žiadostí o pripojenie (funkcia `listen`).
d) V kóde servera doplňte akceptáciu žiadosti o pripojenie (funkcia `accept`). Za akceptáciu doplňte výpis oznamujúci vytvorenie spojenia.
e) V kóde klienta doplňte vytvorenie soketu v UNIX domain (funkcia `socket`).
f) V kóde klienta doplňte spojenie so serverom (funkcia `connect`).
g) Otestujte či server akceptuje pripojenie klienta.
h) Pomocou príkazu `ls -l` zistite, či server vytvoril v pracovnom adresári soket.
i) V kóde klienta doplňte poslanie údajov o študentovi serveru (funkcia `write`, alebo `send`). V kóde servera doplňte prijatie údajov (funkcia `read`, alebo `recv`). Otestujte komunikáciu.
j) V kóde servera doplňte poslanie zmenených údajov klientovi.
k) V kóde servera doplňte uzavretie soketu (funkcia `close`).
l) V kóde servera doplňte odstránenie soketu zo súborového systému (funkcia `unlink`).
m) V kóde klienta doplňte prijatie zmenených údajov zo servera.
n) V kóde klienta doplňte uzavretie soketu.