Լաբորատոր աշխատանք 2

Միջպրոցեսային հաղորդակցություն (IPC)
Հաղորդագրությունների հերթեր (Message Queues)

msgget(), msgsnd(), msgrcv() ֆունկցիաները

Դիտարկվում են msgget(), msgsnd(), msgrcv() ֆունկցիաների պարամետրերի հնարավոր արժեքներն ու արդյունքում ձևավորվող հնարավոր սխալները: Բոլոր ծրագրերն անհրաժեշտ արժեքները ստանում են հրամանային տողի պարամետրերի տեսքով:

1. **message_create.c**
   Ստեղծում է (ստանում է) հաղորդագրությունների հերթ:
   Հրամանի համառոտագիրը.
   ./message_create [-cx] {-f pathname | -k key | -p} [octal-perms]

   Պարամետրերն ունեն հետևյալ նշանակությունը.
   - -c
     Տեղադրել IPC_CREAT դրոշակը
   - -f pathname
     Օգտագործել ftok() ֆունկցիան
   - -k key
     Օգտագործել key արգումենտի արժեքը որպես բանալի
   - -p
     Որպես բանալի օգտագործել IPC_PRIVATE հաստատունը
   - -x
     Տեղադրել IPC_EXCL դրոշակը

   Ծրագրի գործարկման օրինակ.

   ./message_create –c –p 700

2. **message_send.c**
   Հաղորդագրությունների հերթում տեղադրում է հերթական հաղորդագրությունը:
   Հրամանի համառոտագիրը.
   ./message_send [-n] msqid msg-type [msg-text]

   -n պարամետրի առկայության դեպքում տեղադրում է IPC_NOWAIT դրոշակը:

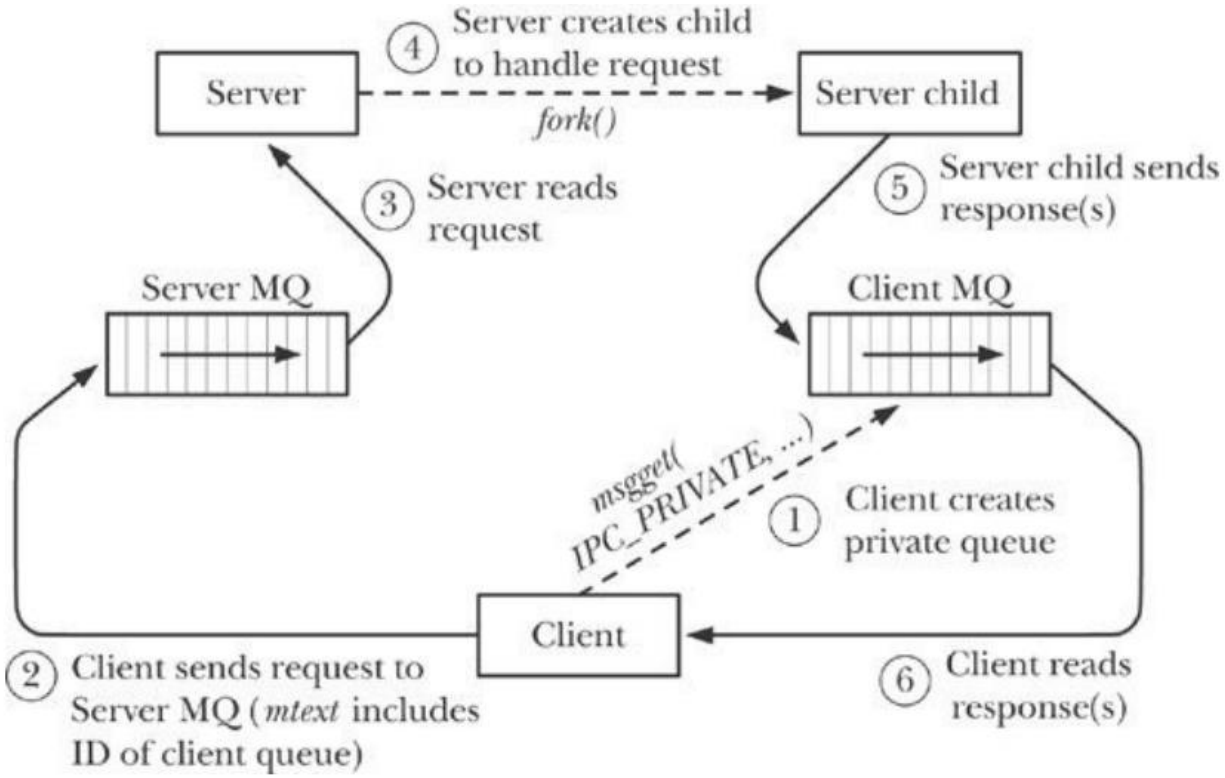
   Ծրագրի գործարկման օրինակ.
   ./message_send –n 0 10 Hello

3. **message_receive.c**
   Հաղորդագրությունների հերթից ստանում է հերթական հաղորդագրությունը:
   Հրամանի համառոտագիրը.
   ./message_receive [options] msqid [max-bytes]

   Պարամետրերն ունեն հետևյալ նշանակությունը.
   - -e
     Տեղադրել MSG_NOERROR դրոշակը
   - -n
     Տեղադրել IPC_NOWAIT դրոշակը
   - -t type
     Նշել ստացվող հաղորդագրության տիպը
   - -x
     Տեղադրել MSG_EXCEPT դրոշակը

   Ծրագրի գործարկման օրինակ.
   ./message_receive –n 0 1024

Client-Server ճարտարապետությունը



Նկ. 1 Client-Server համակարգի կառուցվածքը

Client-Server ճարտարապետությունն օգտագործվում է սերվերի և հաճախորդների (client) միջև տվյալներ փոխանակելու համար: Որպես տվյալների փոխանակման մեխանիզմ կարող է օգտագործվել հաղորդագրությունների հերթը: Դիտարկենք Նկ. 1-ում պատկերված սխեման, որտեղ սերվերի և հաճախորդներից յուրաքանչյուրի համար օգտագործվում է առանձին հաղորդագրությունների հերթ: Հաճախորդներից տվյալներ ստանալու նպատակով սերվերը աշխատանքը սկսելից ստեղծում է նոր հաղորդագրությունների հերթ (**Server MQ**), որի բանալին հայտնի է բոլոր հաճախորդներին: Սերվերից տվյալներ ստանալու նպատակով յուրաքանչյուր հաճախորդ աշխատանքը սկսելիս ստեղծում է նոր հաղորդագրությունների հերթ (**Client MQ**): Սերվերին տվյալներ ուղարկելիս հաճախորդը հաղորդագրության մեջ տեղադրում է իր Client MQ ID-ն, որը սերվերն օգտագործում է պատասխան հաղորդագրություն ուղարկելիս:

Նկատենք, որ սերվերը նախատեսված է տարբեր հարցումներ միաժամանակ սպասարկելու համար (**կոնկուրենտ սերվեր**): Այդ նպատակով հաճախորդից հարցում ստանալիս սերվերի հիմնական պրոցեսը ստեղծում է դուստր պրոցես (fork()), որին փոխանցում է հարցման սպասարկումը, իսկ հիմնական ծրագիրը շարունակում է սպասել նոր հարցումների: Կոնկուրենտ սերվերի այլընտրանքը **իտերատիվ սերվերն** է, որը չի կարող միաժամանակ տարբեր հարցումներ սպասարկել, քանի որ բոլոր հարցումների սպասարկումն իրականացվում է մեկ պրոցեսի կողմից:

Նման համակարգի օրինակ ներկայացված է հետևյալ ծրագրերում.

- **msg_file.h**
  Սահմանում է հաճախորդից սերվերին ուղարկվող (requestMsg) և սերվերից հաճախորդին ուղարկվող (responseMsg) հաղորդագրությունների կառուցվածքը:
- **msg_file_server.c**
  Ստեղծում է սահմանված հայտնի բանալիով (SERVER_KEY) հաղորդագրությունների հերթ (Server MQ) և կոնկուրենտ կերպով իրականացնում հարցումների սպասարկում. հաճախորդից Server MQ-ի միջոցով ստանում է ֆայլի անվանում և վերադարձնում է այդ ֆայլի պարունակությունը հաճախորդին Client MQ-ի միջոցով:
- **msg_file_client.c**
  Ստեղծում է նոր հաղորդագրությունների հերթ (Client MQ) և սերվերին փոխանցում հաղորդագրություն, որի մեջ գտնվում են ֆայլի անվանում և նոր ստեղծված հերթի id-ն: Նշված հերթի id-ն օգտագործվելու է սերվերի կողմից պատասխան հաղորդագրություն ուղարկելու համար:

Առաջադրանքներ

1. Ստեղծել նոր հաղորդագրությունների հերթ՝ օգտագործելով message_create ծրագիրը և msgget ֆունկցիային որպես բանալի փոխանցելով IPC_PRIVATE

հաստատունը: Բացատրել ծրագրի հաջորդական կատարումների արդյունքում ստացվող հաղորդագրությունները:

2. Ստեղծել նոր հաղորդագրությունների հերթ՝ message_create ծրագրին փոխանցելով **–k** արգումենտը: Բացատրել –k և –p արգումենտների տարբերությունը:
3. Ստեղծված հերթերից մեկում message_send ծրագրի միջոցով տեղադրել 3 նոր հաղորդագրություն, որոնք կունենան հետևյալ դաշտերը.
    - mtype = 10, mtext = "message 1"
    - mtype = 20, mtext = "message 2"
    - mtype = 25, mtext = "message 3"
4. Վերը նշված հերթից կարդալ mtype = 20 ունեցող հաղորդագրությունը՝ օգտագործելով message_receive ծրագիրը:
5. Դատարկել հերթը՝ կարդալով առկա բոլոր հաղորդագրությունները: Այնուհետև կատարել կարդալու ևս մեկ փորձ՝ message_receive ծրագրին փոխանցելով –n արգումենտը: Բացատրել ծրագրի աշխատանքի արդյունքը:
6. Կատարել msg_file_server և msg_file_client ծրագրերը: Բացատրել աշխատանքը: