

ОС лаба 1

Задание 1

Парные сокеты, `sockpair`

Создаем `child` и он с `parent` обменивается сообщениями – дуплексная связь

`Child` посылает “Hello parent”, и `parent` посылает ...

Задание 2

Взаимодействие через сокеты на отдельно стоящей машине `AF_UNIX`, Type – `DGRAM`

2 варианта

- а) Клиент только посылает сообщение серверу, ничего не получает от сервера
- б) Взаимодействие клиента и сервера. Клиент посылает сообщение-запрос, сервер посылает сообщение-ответ

Реализуем КАЛЬКУЛЯТОР

Клиент посылает 2 числа и операцию. Сервер должен посчитать

2 программа:

Клиент отправляет серверу 2 числа и операцию

Сервер выполняет работу калькулятора

На стороне клиента, т.к. клиент получает от сервера ответ, надо заполнить поля `struct sockaddr`. Как правило в качестве файла клиента указывается `PID.cli`

Надо вызывать `bind`, чтобы связать клиента с его адресом

Посылка – `sendto` с указанием адреса клиента. Но для получения ответа надо вызвать `recvfrom`. Никаких `connect`

На сервере: инициализируются поля адреса сервера, устанавливается связь, `recvfrom(адрес клиента)`, `sendto`

Сист. вызов `bind` связывает файл дескриптор сокета сервера с адресом сервера. Имя сервера задается `#define`

Вывод как в RPC, на стороне клиента: $a + b = c$

Когда демонстрируем запуском много клиентов. Можно создать клиента с циклом повторения обращения к серверу. Можно `fork`

Нужно ли здесь взаимное исключение

`SIGTERM`

Задание 3

Листинг 5.9, стр 171

Один и тот же клиент работает как читатель и как писатель.

Как читатель – получает информацию от сервера (массив)

(элементы массива могут быть недоступны, они помечаются как пробел или -1)

Клиент на основе данных массива запрашивает свободный элемент и посылает индекс.

Клиент запросил соединение, получил и отправил признак что он читатель, сервер отправляет ему массив.

После этого клиент переходит в режим записи и указывает индекс массива который он хочет получить.

Тот же самый процесс, который обслуживал его как читателя будет обслуживать его как писателя

Запись в массив только в режиме монопольного доступа

Потом на потоках, у потоков нет собственного АП

Мультиплексирование

Сетевой сервер в 3х вариантах, кот работает по задаче читатель-писатель:

На сервере есть массив (или вектор)

Клиент может обращаться к серверу как читатель или как писатель

Если клиент обращается как читатель, т е он посылает букву 'r', то сервер предоставляет ему возможность прочитать этот массив, то сервер должен его послать. Сервер посылает массив, в этом массиве какие-то элементы уже могут быть заняты, т е он ему посылает текущее состояние массива

Если ячейка пустая (элемент массив – ' ' или -1), то это – элемент недоступен.

Клиент может выбрать элемент (его индекс), указать что он писатель и это уже должно выполняться в режиме монопольного доступа. Если окажется что др клиент оказался шустрее, то возвращается – 'элемент занят'. Если никто не претендует, то в режиме монопольного доступа сервер определяет этот элемент как занятый и заменяет значение элемента на пробел или -1 и отсылает что ОК.

3 варианта:

- а) 1 асепт и распараллелить fork() (в книге есть пример)
- б) 1 асепт и распараллелить pthread() (надо уточнить про id и про группу)
- с) Мультиплексирование (на лек 2 будет 2 модели ввода-вывода – разговор про мультиплексирование)

Читатель читает индекс, выбирает тот который свободен, переходит в режим писателя

Один и тот же клиент

Мультиплексирование

сравнить время ответа сервера при многопоточной реализации и при многопоточной
на сервере не надо sleep
реальное время ответа
сравнить

выбрать как мерить задержку, компьютеры работают быстро => может надо мерить в
тиках

мультиплексер для того чтобы потом обрабатывать пришедшие соединения создается
массив, т е обработка соединений будет выполняться по очереди!

сервер однопоточный

калькулятор

важно! получить реальное время ответа сервера на наборе! клиентов
на сервере не надо задержки!

Нужно взять программу с pthread и переделать сервер на мультиплексоры с epoll. То
есть по сути получается один клиент и 2 вида сервера.

Потом надо сделать замеры времени обращения к серверу и для pthread и для
мультиплексора (причем замеры там можно делать как на стороне клиента так и на
стороне сервера) и

она хочет получить 2 колоночки где в одном столбике замеры pthread а рядом
мультиплексора и в конце подсчет среднего времени.

Однопоточный но делаем gw

Просто обслуживание в порядке очереди должно получаться

Насколько помню там она говорила замерять перед accept, в pthread от pthread create
до pthread detach

Там тип старт замеров делаешь перед accept, а конец после обработки recv..

Вообще я на семинаре за ней записала „мультиплексирование будет эффективнее”

Но на лабе потом она другое говорила

И вообще, у кого-то вышло время +- одно и то же

И тогда ответ такой : если время +- одинаковое, то предпочтительное

мультиплексирование, так как оно менее затратное

Чтоб время заседать надо соответственно все слипы убрать и сделать чтоб побольше
работало, поэтому массив с буквами должен несколько раз по кругу заполняться