Client:

1. 透過 semget 開啟 server 端建立的 semaphore。
2.透過 socket()建立使用 IPv4 協定及 TCP 協定 的 socket client，接著利用 connect()，將 client 連接至 server。
3.根據使用者提供的資訊進行動作(deposit/withdraw、金額、次數)。
4.當每次進行動作時呼叫 P()將 binary semaphore 減一(設為 0)以進行資源的保護(避免多個 client 同時傳遞資料到 server 造成 race condition)，當次動作完成時，呼叫 V()將 binary semaphore 加一(設為 1)，允許其他 process 進行動作。

Server:

1.首先透過 semget 建立 semaphore，接著透過 semctl()將 semaphore 初值設為一，以供其他 process 進行操作。
2.透過 socket()建立使用 IPv4 協定及 TCP 協定 的 socket server，利用 setsockopt() 強制使用已在使用 socket address，利用 bind()將前面創建的 socket 綁訂到指定的 port 上，而 ip address 則是透過 INADDR_ANY 設定為可連接任何 ip address。接著透過 listen() 將欲連線者 排入 queue 中。
3.利用 accept()從 queue 取出已連線的 socket，並配合 pthread_create()為每個已連線的 client 建立執行序，使得 server 能夠與多個 client 同時進行連線，而 race condition 的問題已在 client 端透過 semaphore 解決。
4.透過 ipcs -a 可以看到該 semaphore 的資訊，按下 Ctrl＋C 透過 signal 呼叫 semctl(s,0,IPC_RMID,0) 以清除 semaphore，再次使用 ipcs -a 即可發現該 semaphore 已被清除。

重點:

1.使用 binary semaphore 實現 Single Shared-Resource-Access，保護資源避免發生 race condition 問題。
2.使用 pthread 使 server 端能夠支援同時多人連線。