TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN HCM
Khoa Công nghệ thông tin

BÁO CÁO BÀI TẬP NHÓM 01

WEB SERVER

GVHD: Nguyễn Thanh Quân
Lê Hà Minh

MSSV: 1312232
Họ và Tên: Đặng Bá Quang Huy
Email: quanghuy4994@gmail.com
SĐT: 0972678420

MỤC LỤC

I. Bảng Phân Công Công Việc

Công Việc	Người Thực Hiện	Ghi chú
Xây dựng web server xử lí theo cơ chế tuần tự (iterate)	Đặng Bá Quang Huy	
Xây dựng web server xử lí theo cơ chế đa tiến trình (process)	Đặng Bá Quang Huy	
Xây dựng web server xử lí theo cơ chế đa tiểu trình (thread)	Đặng Bá Quang Huy	
Xây dựng web server xử lí theo cơ chế non-blocking (select)	Đặng Bá Quang Huy	
Xây dựng web server xử lí theo cơ chế non-blocking (poll)	Đặng Bá Quang Huy	
Xây dựng web server xử lí theo cơ chế non-blocking (epoll)	Đặng Bá Quang Huy	

II. Mô tả hàm, chức năng chính

Tên Hàm	Tham Số	Chức Năng	Kết quả trả về
▪ startup	*port: kiểu char, chứa port để server hoạt động multisocket: kiểu char, bằng 0 là tắt multisocket, bằng 1 là bật multisocket	Hàm này được sử dụng để khởi tạo các tham số cần thiết cho server lắng nghe trên port được truyền vào	Biến kiểu int giữ socket mà server đang lắng nghe kết nối.
▪ main	*argc: kiểu int chứa số tham số truyền vào *argv: mảng char** chứa giá trị của các tham số truyền vào	Hàm chính của chương trình. Gọi thực hiện các kiểu xử lí khác nhau như poll, epoll, select, iterate…	
▪ accept_request	lclient: kiểu int, giữ socket của client cần nhận dữ liệu gửi lên	Nhận dữ liệu từ client gửi lên và đưa vào xử lí => gửi kết quả cho client	Biến kiểu int là kết quả của hàm. Nếu là -1 là có lỗi xảy ra
▪ recv_request	lclient: kiểu int, giữ socket của client cần nhận dữ liệu gửi lên	Nhận dữ liệu từ client gửi lên và đưa vào xử lí.	struct data gồm: query: chứa lệnh truy vấn result: chứa kết quả trả về.

			-1: Lỗi 1: Trả về file 2: Trả về kết quả truy vấn
▪ exec_query	client: kiểu int, giữ socket của client cần gửi kết quả trả về query: kiểu const char* chứa chuỗi truy vấn của client	Thực thi truy vấn và gửi kết quả truy vấn cho client	Không có
▪ send_file	client: kiểu int, giữ socket của client cần gửi header *filename: kiểu const char* chứa tên của file cần gửi	Đọc nội dung file từ server và gửi cho client	Không có
▪ header	client: kiểu int, giữ socket của client cần gửi header ct_len: kiểu int, chứa content-length của gói tin	Sử dụng để tạo header và gửi về cho client	Không có
▪ file_size	*filename: kiểu const char* chứa tên của file cần tính kích thước	Sử dụng để tính kích thước file (đưa vào header) trước khi gửi cho client	Biến kiểu int là kích thước của file cần tính
▪ get_type	*filename: kiểu const char* chứa tên của file cần kiểm tra kiểu *content type: kiểu char* chứa content-type sau khi xử lí	Trả về content-type của file để gắn vào header tương ứng	Không có
▪ init_capitals		Khởi tạo đối tượng json đọc từ file capital.json để lấy dữ liệu cho xử lí	Trả về một con trỏ struct json-object* để truy vấn dữ liệu

III. Chạy và kết quả nhận được

- Gọi chạy chương trình bằng lệnh
  ./1312232 -<iterate>|<process>|<thread>|<select>|<poll><epoll>

Trong đó các tham số

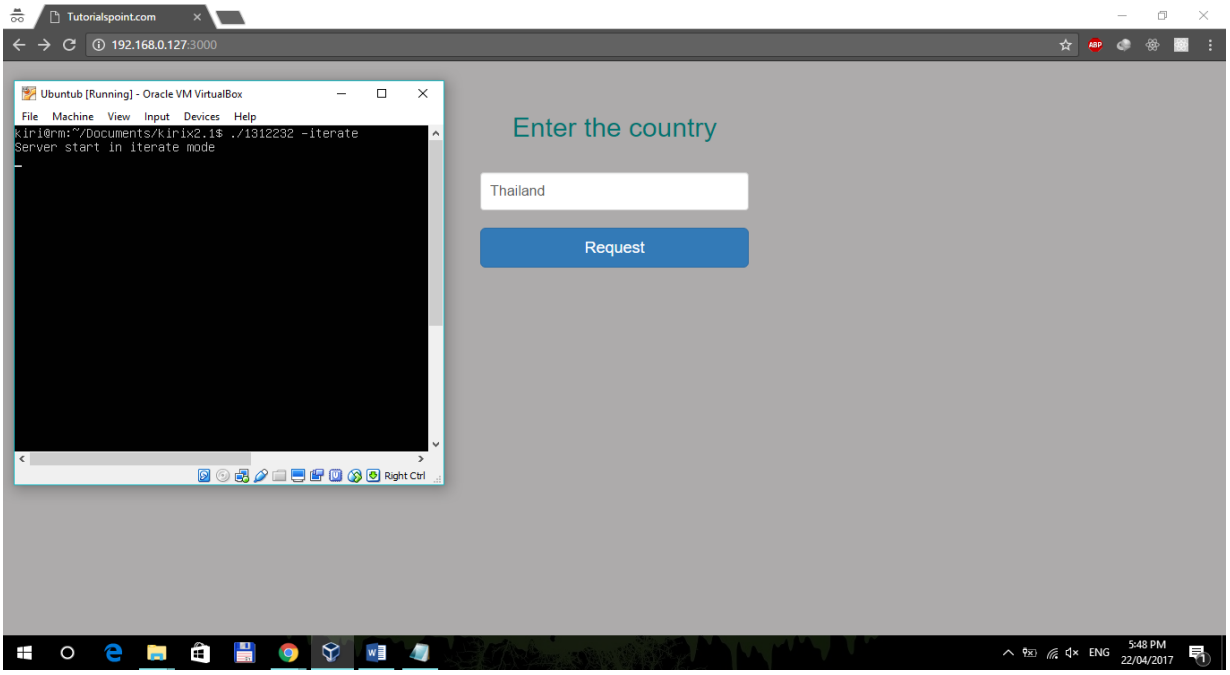
- iterate: Xử lí request tuần tự
- process: Xử lí request bằng cách tạo nhiều tiến trình song song
- thread: Xử lí request bằng cách tạo ra nhiều tiểu trình chạy song song
- select: Sư dụng cơ chế non-blocking
- poll: Sử dụng cơ chế non-blocking cải tiến của select

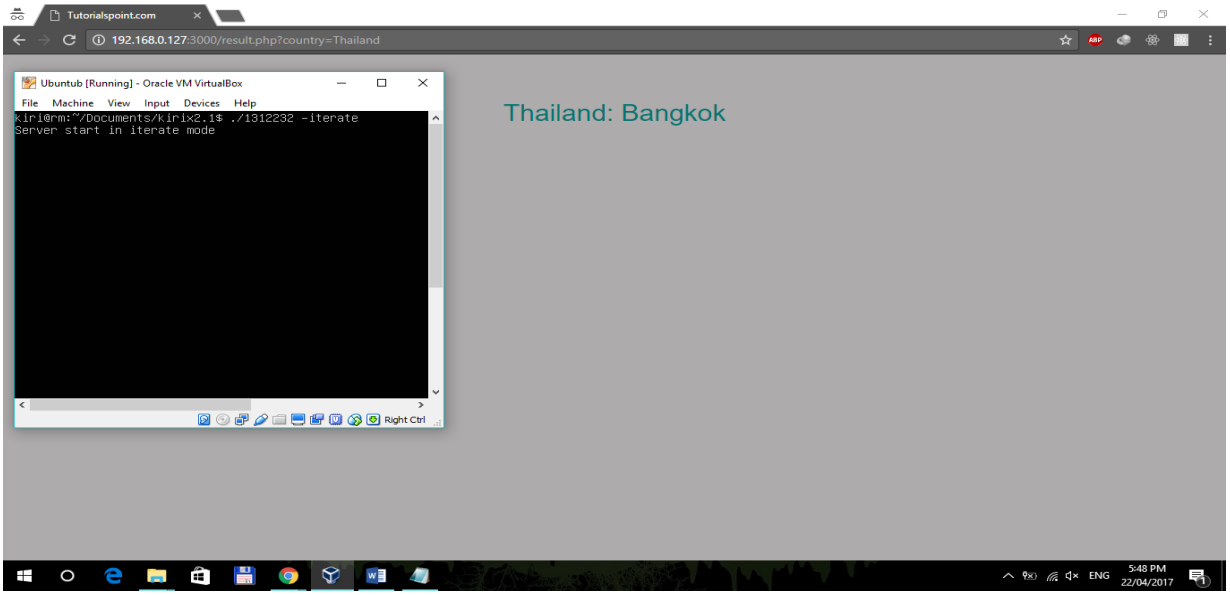
- epoll: Sử dụng cơ chế non-blocking (tối ưu)

1 Xử lí tuần tự

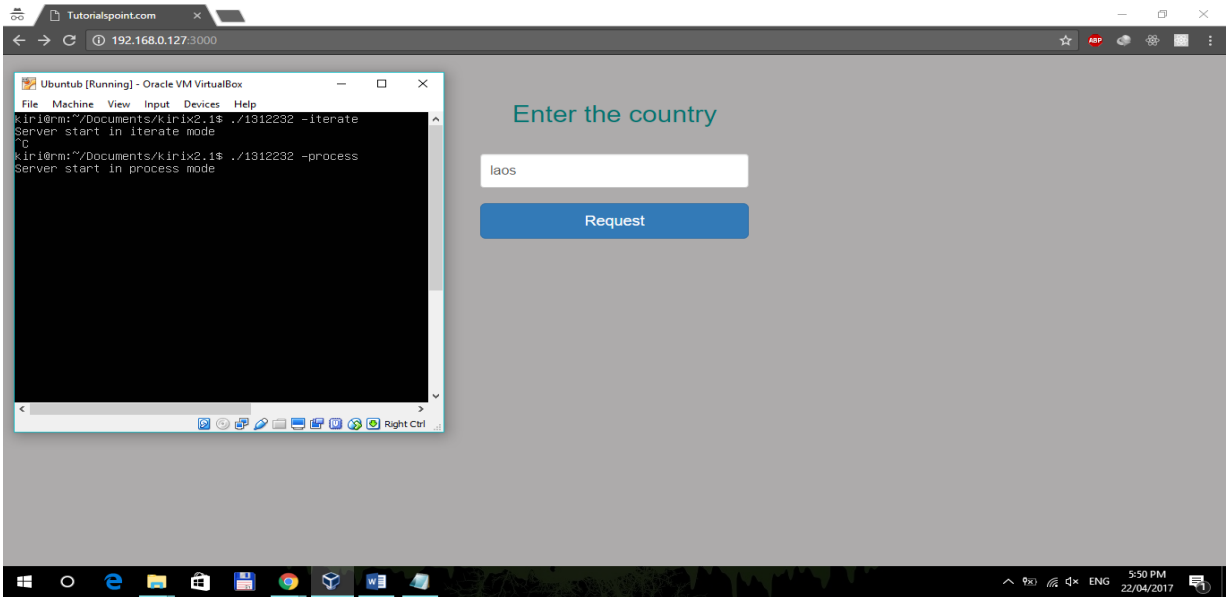
```
kiri@rm:~/Documents/kirix2.1$ ./1312232 -iterate
Server start in iterate mode
```

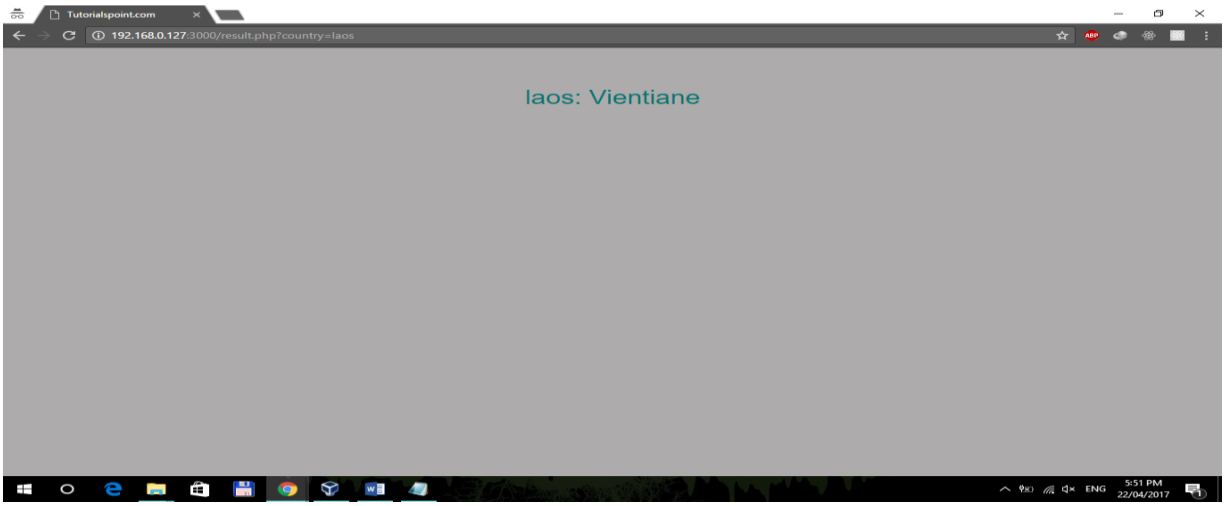
Kết quả:



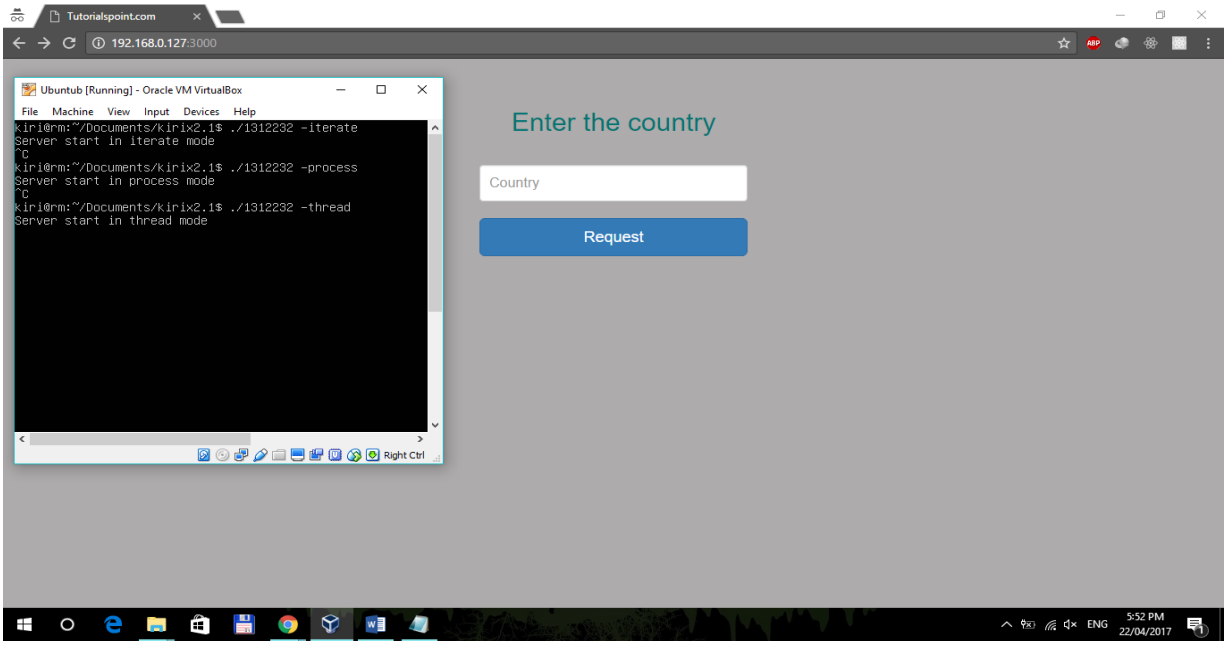


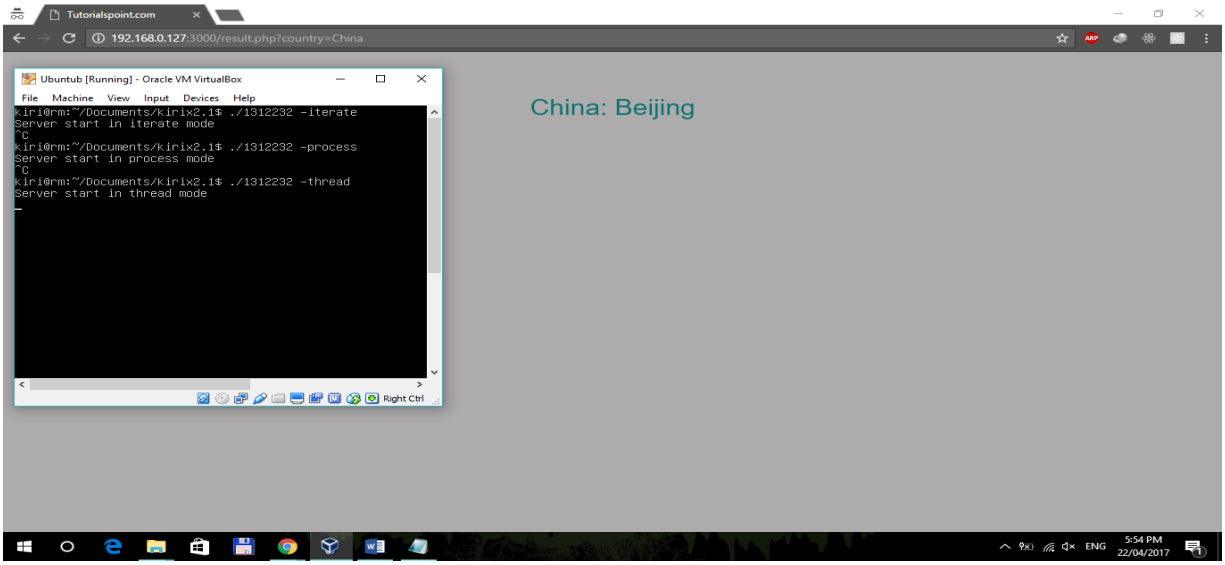
2 Xử lí bằng đa tiến trình



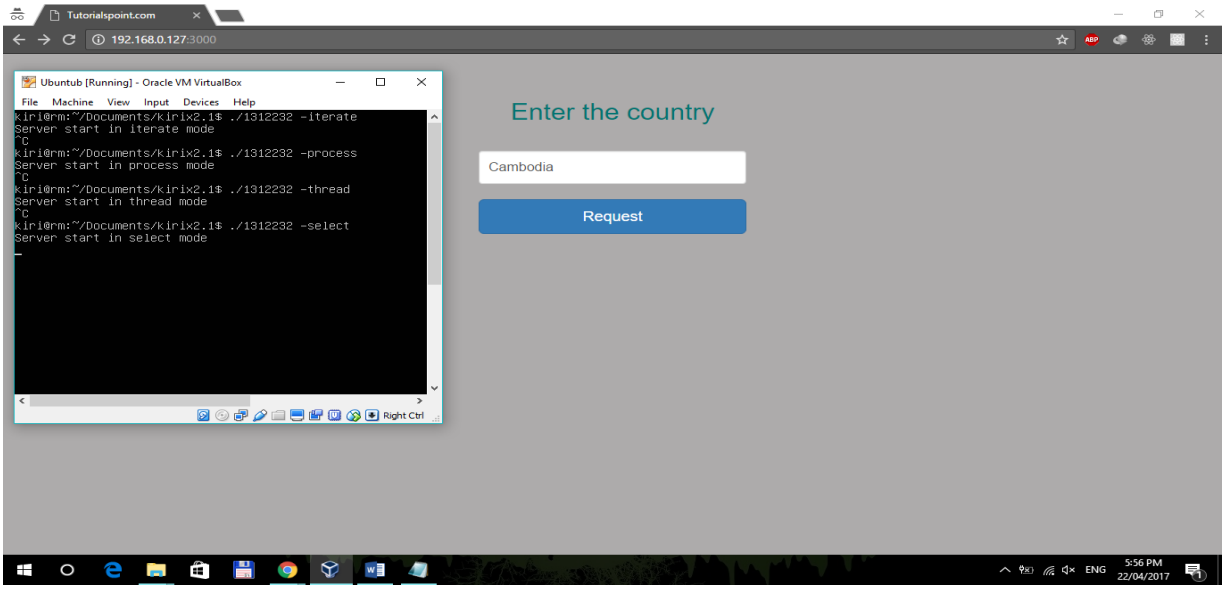


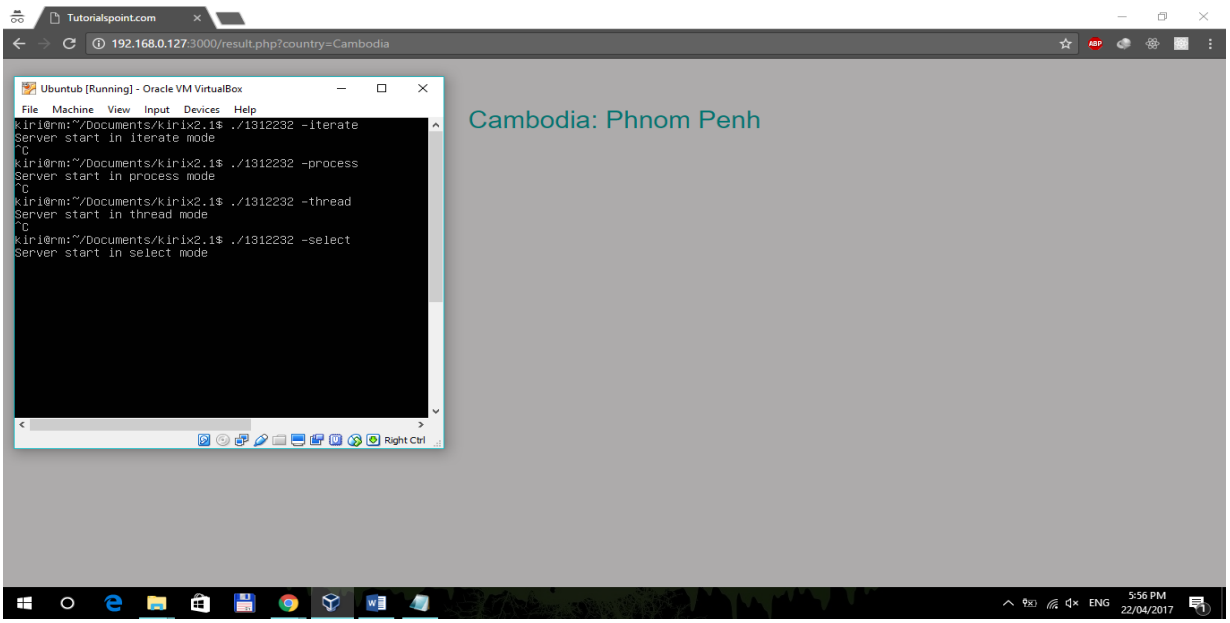
3 Xử lí bằng đa tiểu trình



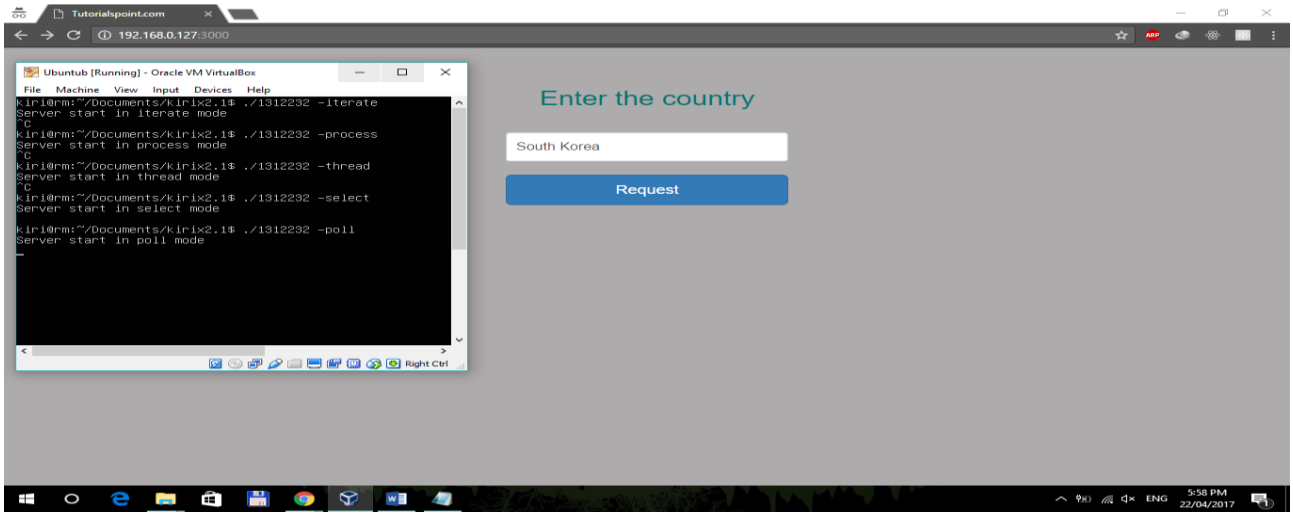


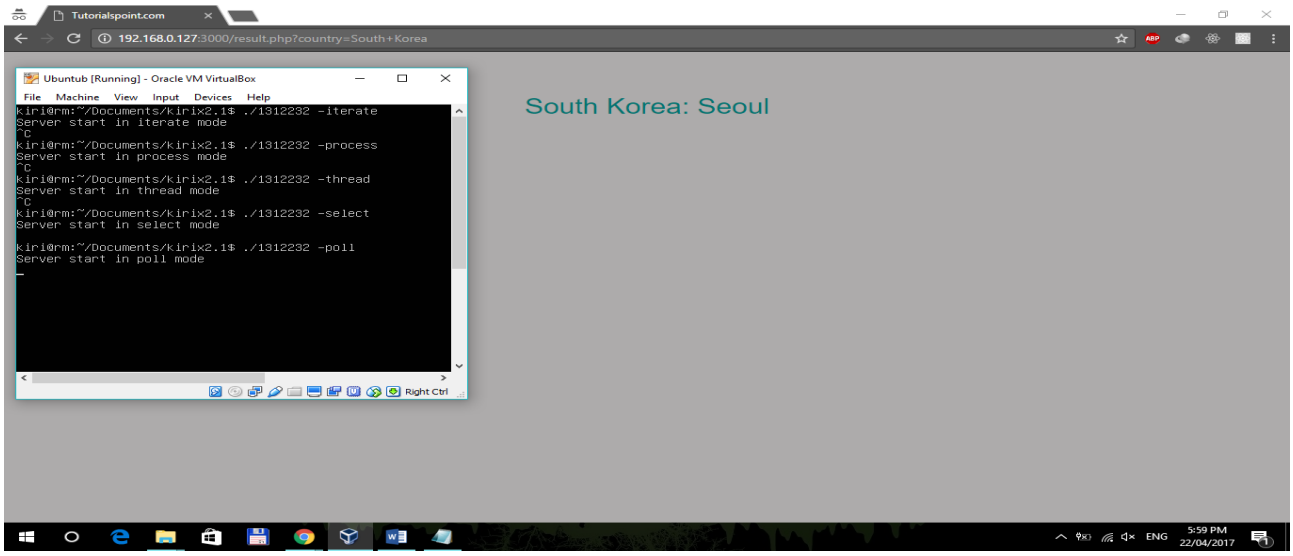
4 Xử lí bằng select



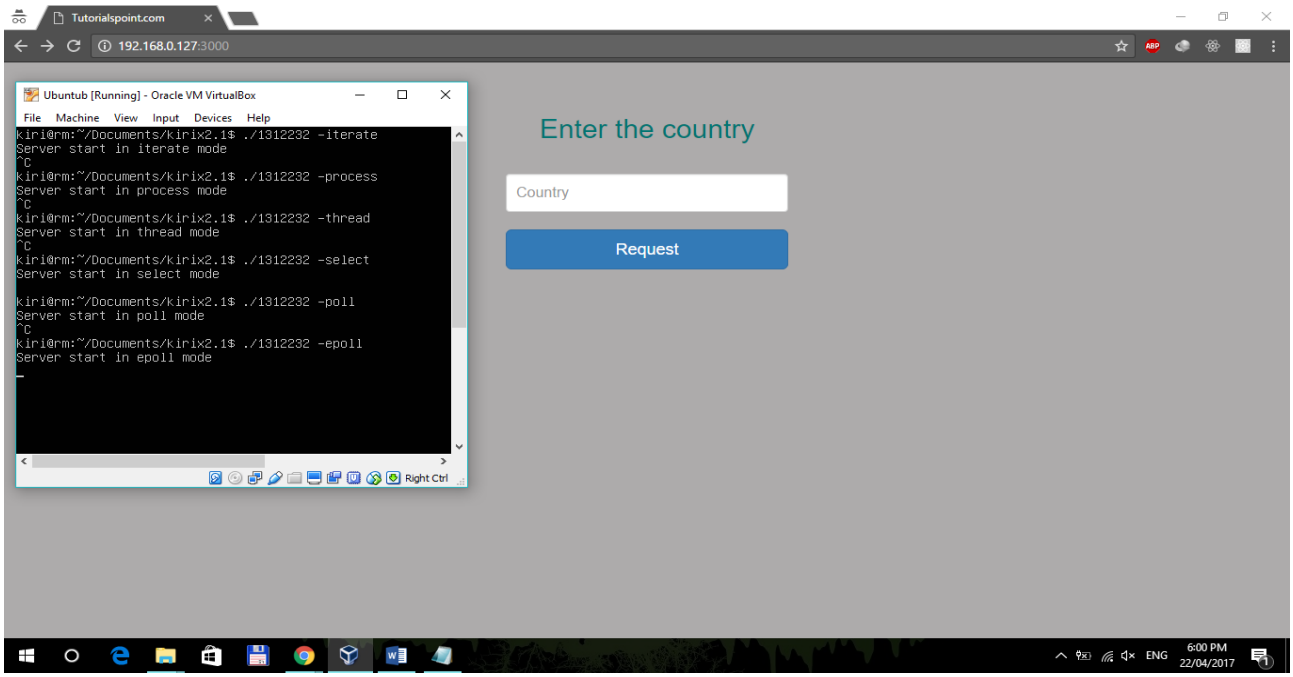


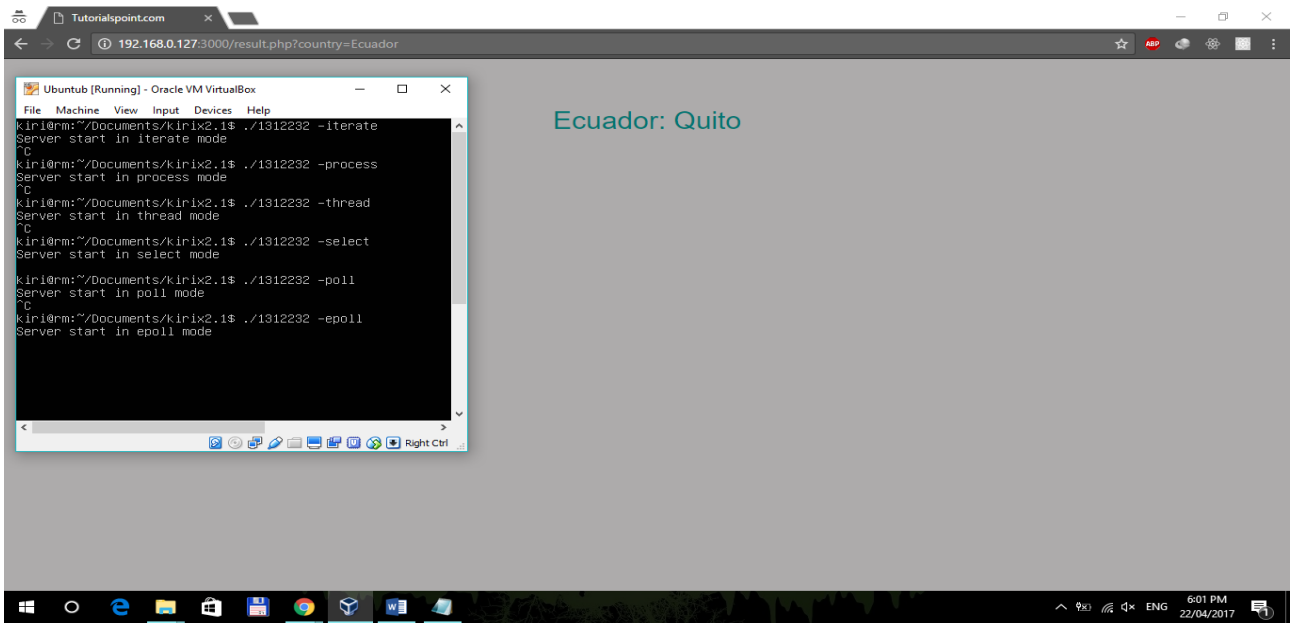
5 Xử lí bằng poll





6 Xử lí bằng epoll



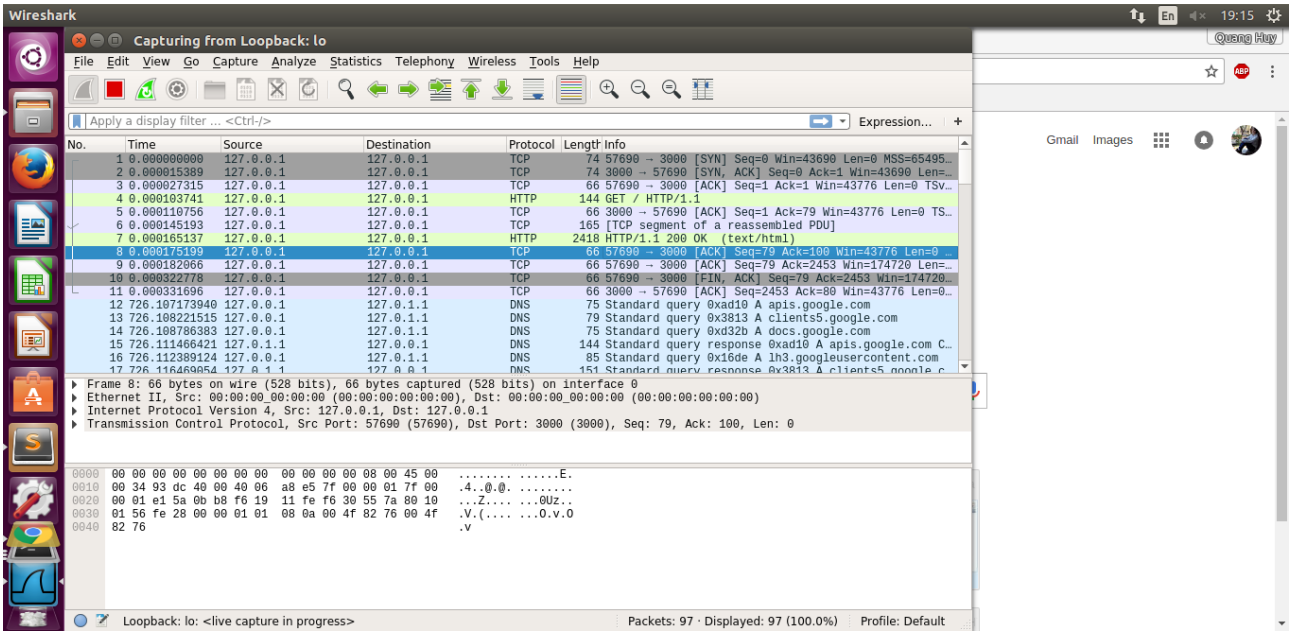


IV. Công việc đã làm và mức độ hoàn thành

Công Việc	Người Thực Hiện	Tiến Độ	Ghi chú
Xây dựng web server xử lí theo cơ chế tuần tự (iterate)	Đặng Bá Quang Huy	100%	Hoạt động chậm khi có nhiều kết nối
Xây dựng web server xử lí theo cơ chế đa tiến trình (process)	Đặng Bá Quang Huy	100%	Dễ chết khi có nhiều kết nối (phụ thuộc tài nguyên hệ thống)
Xây dựng web server xử lí theo cơ chế đa tiểu trình (thread)	Đặng Bá Quang Huy	100%	Dễ chết khi có nhiều kết nối (phụ thuộc tài nguyên hệ thống, bị giới hạn bởi hệ điều hành)
Xây dựng web server xử lí theo cơ chế non-blocking (select)	Đặng Bá Quang Huy	100%	Tốc độ nhanh. ổn định. Nhưng bị lỗi khi số lượng kết nối quá nhiều 16s (3000*10 request)
Xây dựng web server xử lí theo cơ chế non-blocking (poll)	Đặng Bá Quang Huy	100%	Tốc độ nhanh và ổn định hơn select 12s (3000*10 request)
Xây dựng web server xử lí theo cơ chế non-blocking (epoll)	Đặng Bá Quang Huy	100%	Tốc độ nhanh và ổn định nhất 8s (3000*10 request)

V. Mô tả quá trình gửi nhận dữ liệu

1 Ảnh chụp quá trình gửi & nhận gói tin (wireshark)



2 Sơ đồ gửi và nhận gói tin

B1: client gửi gói SYN đến web server yêu cầu kết bối

B2: web server gửi goisACK/SYN đến cho client chấp nhận kết nối

B3: client gửi lại cho web server gói tin ACK chấp nhận kết nối

B4: client gửi gói tin HTTP (GET / HTTP/1.1) yêu cầu lấy trang / ở web server

B5,B6,B7: web server gửi lại gói tin ACK đã nhận được gói tin và chấp nhận yêu cầu. Đồng thời gửi lần lượt 2 gói tin chứa header và body của trang / (“index.html”)

B8, B9: client gửi 2 gói tin ACK thông báo đã nhận được 2 gói (header và body)

B10: sau khi nhận đủ gói tin. Client gửi gói tin FIN lên yêu cầu web server ngắt kết nối.

B11: web server gửi lại gói tin ACK. Báo đã nhận được yêu cầu và chấp nhận ngắt kết nối