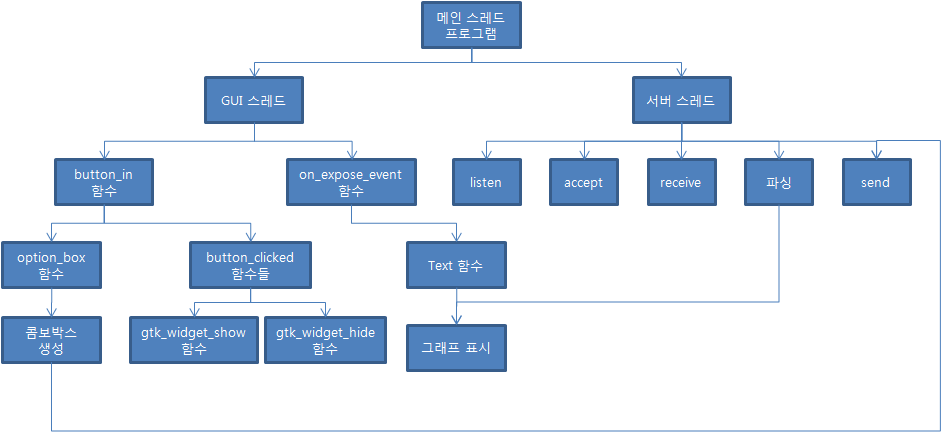
**통합 개발 문서**

**통합 서버 전력 관리 프로그램(Server Mark v0.8)**

**Software 내용**



**<그림 3. 관리자 프로그램 메뉴 구성도>**

서버 마크 관리자는 위와 같이 멀티 스레드 인터페이스이다.

# 정보

* GUI 스레드 - GUI 부분을 담당하는 스레드로 화면에 표시되는 모든 화면을 담당 한다.
  + on\_expose\_evet 함수 – 화면에 그래프와 드라이버 옵션 메뉴의 글씨를

그려주는 역할로 하위에 text 함수가 있다.

* + - text함수 – 글씨를 받아와 정해진 위치 글씨체로 뿌려주는 역할
  + button\_in 함수 – 버튼을 생성하고 신호를 연결시켜 주는 함수부로 버튼 위젯   
    생성 라벨 연결, 신호 연결 등의 동작을 수행한다.
    - option\_box 함수 – 드라이버 설정에서 사용하는 콤보 박스를 만들어 주는   
      함수로 콤보 박스를 생성하고 옵션 값 들을 연결시켜 주는 역할을 수행

하고 콤보 박스 변경시 서버 부의 send 함수를 호출하여 콤보 박스의 값을 넘겨준다.

* + - button\_clicked 함수 – 버튼 클릭시 호출되는 함수 부로 버튼 마다 필요한 값들을 설정하여 주고 gtk\_widget\_show와 gtk\_widget\_hide 함수를 이용하여 콤보박스의 숨김과 표시 동작을 수행하여준다.
* 서버 스레드 - 서버를 담당하는 스레드로 서버를 생성하고 측정값을 받아와 파싱과 GUI의옵션 전달시 send 역할을 담당한다
  + 서버부 세부 함수들은 기본 TCP/IP통신의 함수들을 사용하므로 생략하고 파싱부만 설명한다.
  + 파싱부 – recv함수에서 값을 받게 되면 그 값을 파싱하여 변수에 저장시켜 준다. 값의 자릿수를 맞추기 위해 스택을 이용하여 뒷자리부터 계산하여 변수에 넣게 된다.

# 프로그램 개발 중 문제점 및 해결 사례

1. 프로그램 작성시 cairo와 gtk 위젯과의 연동중 표현 우선순위 관련 에러
   1. 본 프로그램을 작성하면서 cairo와 gtk를 처음 접하였기 때문에 gtk와 cairo와의 우선권이 모호해 생긴 문제
      1. 해결방법 – gtk의 우선권이 더 높기 때문에 cairo를 background 에 배정하고 그 위에 gtk위젯들을 배치하여 문제를 해결
   2. 드라이버 설정시 버튼의 숨김 표시 문제.
      1. 해결방법 – gtk\_widget\_show 함수를 이용하여 위젯의 표시를 각각 조정하여   
         위젯들의 표시 숨김을 조절
   3. 서버와 gui인터페이스 동시 실행 문제
      1. 해결방법 – 서버와 gui인터페이스 모두 연속적으로 서버의 함수와 gui의 draw   
         부 함수를 호출하기 때문에 교착 상태가 발생하게 되므로 서버부와 GUI부의   
         함수를 섞는 방식 보다 깔끔한 멀티 쓰레드를 사용하여 동시에 동작 하도록   
         구성

# 프로그램 사용법

**-설치방법-**

개발 시 서버마크 관리자를 설치한 환경은 우분투 9.0.4이며 데스크탑 버전으로 설치하여 GUI 환경을 사용하였다. 관리자는 배포형태가 아니라 아래와 같은 작업을 수행하여 개발환경을 구축해야 한다.

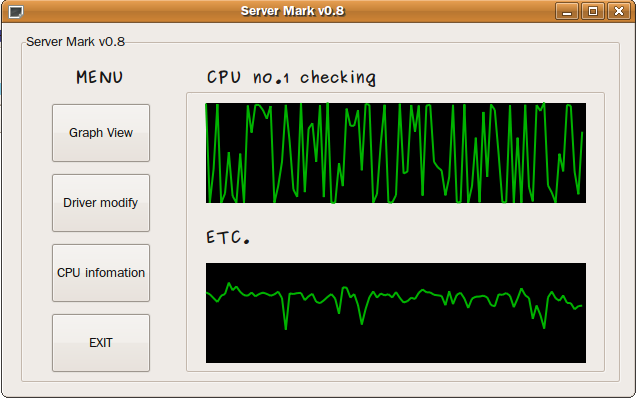
0 build-essential 패키지 설치 : 터미널에서 sudo apt-get install build-essential

0 GTK development 패키지 설치 : sudo apt-get install libgtk2.0-dev

0 카이로 설치 : sudo apt-get install libcairo2-dev

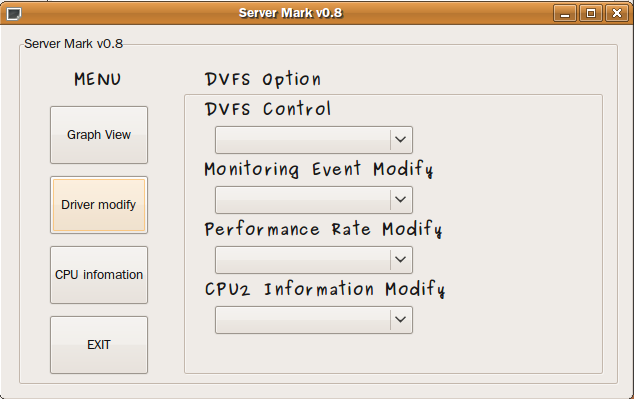
0 GTK+ 와 카이로를 연동하여 컴파일 :

gcc -o 실행파일명 ‘pkg-congif --cflags --libs g사+-2.0’ 소스 파일

**-인터페이스 방법-**

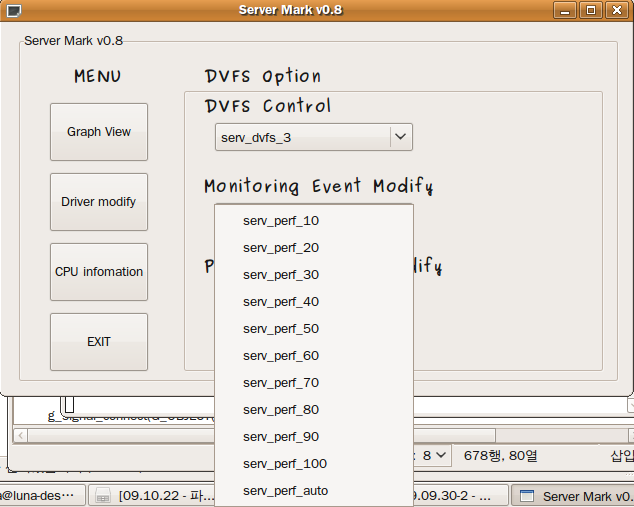
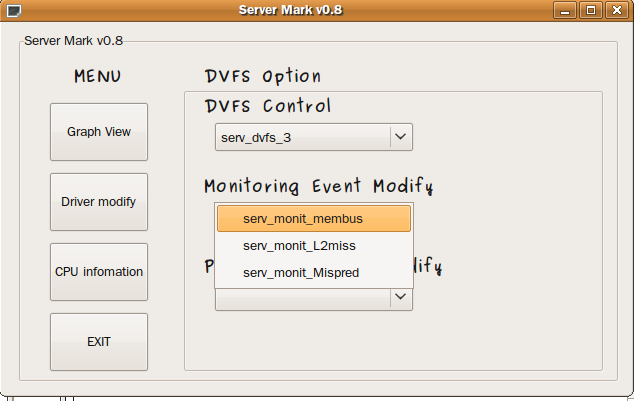
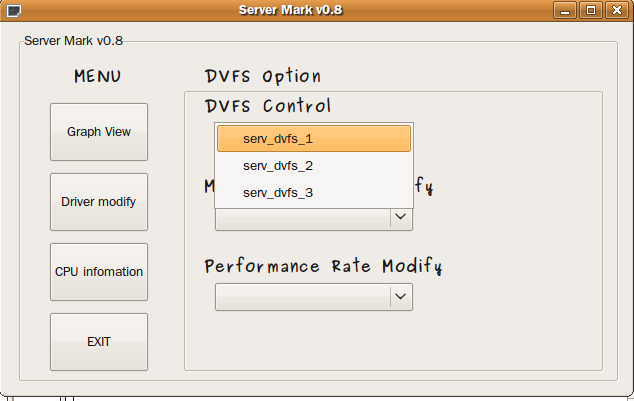
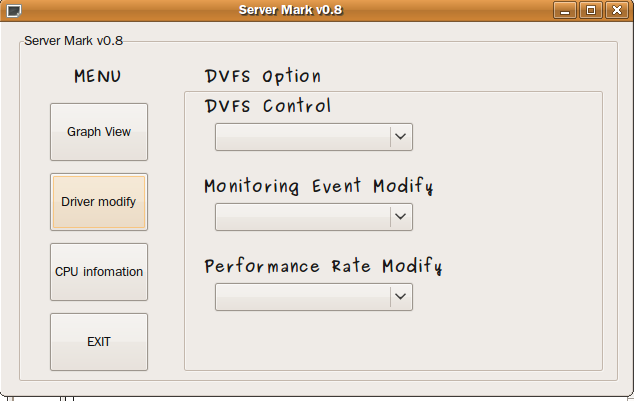
**<그림 4.Graph View 창>**

그림 4는 관리자 프로그램의 Graph view 부분으로 에이전트로부터 수신한 성능 이벤트 값을 표시한다. 위쪽 그래프는 기본적으로 프로세서의 명령어 수행 횟수를 의미하는 instruction retired 이벤트의 값을 출력한다. 아래쪽 그래프는 기본적으로 시스템이 메모리를 엑세스한 횟수를 의미하는 Memory bus rate를 나타내며, Driver modify 메뉴에서 다른 이벤트 값으로 설정할 수 있다.



**<그림 5. Driver modify 창>**

그림 5는 드라이버의 설정을 변경할 수 있는 창으로 DVFS 제어, 모니터링 이벤트 설정, 최소 성능비율을 설정할 수 있다. 그림 6은 각 해당 메뉴의 내용들을 보여준다. 먼저 왼쪽 위의 DVFS Control 부분에서는 원격 서버 PC의 DVFS 단계를 조절할 수 있다. serv\_dvfs\_3 명령이 최고 속도를 나타낸다. 오른쪽 위의 Monitoring Event Modify 부분에서는 Graph View 메뉴의 두 번째 그래프에 출력되는 이벤트를 설정하며, 현재는 momory bus rate, L2 miss rate, misprediction 세 개의 이벤트를 설정할 수 있다. 왼쪽 아래의 Performance Rate Modify 메뉴는 임의로 성능 한계치를 줄 수 있는데 각각의 퍼센티지 비율을 줄 수 있다. 예를 들어 50으로 설정하게 되면 최대 속도에서 성능의 50%의 손실을 감안하는 수준으로 전력관리를 수행하는 것을 의미한다. 벤치마크를 통해서 그림 7과 같은 수준의 전력 소모량 감소 및 전력대비 성능을 확인하였으며, 뛰어난 성능을 보인 것은 그림과 같이 50, 70, auto 항목이다. auto 항목은 시스템의 부하에 따라 자동적으로 DVFS를 적용하는 정책이다.



**<그림 6. Driver modify 창의 각 메뉴 항목>**

**<그림 7. 각 성능비율에 따른 전력감소량 및 전력대 성능비>**



**<그림 8. CPU Information 창>**

CPU Information 창은 에이전트가 설치된 시스템의 CPU 및 관련 시스템 정보를 출력해 주는 부분이다. 아직은 단일 프로세서에서만 동작하기 때문에 구현되지 않았으며, 추 후 다중 플랫폼을 지원하도록 업데이트 시 구현할 예정이다.

# 공통부분

**-사용방법-**

먼저 서버마크 관리자 프로그램을 실행시키면 기본 인터페이스가 나타나며 에이전트의 접속을 대기하게 된다. 다음으로 setup.sh 쉘 파일을 통해 에이전트 프로그램을 실행시키면 서버마크 관리자로 접속을 시도하며, 정상적으로 접속되었을 경우 에이전트가 설치된 시스템의 기본 성능 이벤트 정보 (instruction retired, Memory bus access) 값이 그래프로 출력되기 시작한다.

3-4. 개발환경

- 개발언어 , tool, 사용 솔루션(상용 및 공개SW) 등

사용 또는 개발언어, TOOL

0 서버 : Linux Server(Ubuntu 9.0.4)

0개발언어 : C, GTK++2.0, Cairo

0 TOOL : Ultraedit ver 13.0(정식 버전), 리눅스 터미널(putty)

사용시스템(서버마크 관리자 프로그램 동작)

프로세서 : Intel Pentium4 3.0GHz

메모리 : 2Gbyte RAM

운영체제 : Windows XP 상에서 VMware를 이용하여 Ubuntu 리눅스(9.0.4) 구동

사용시스템(서버마크 에이전트 프로그램 동작)

프로세서 : Intel Core2Duo T7200

메모리 : 2Gbyte RAM

운영체제 : Ubuntu 리눅스(9.0.4)

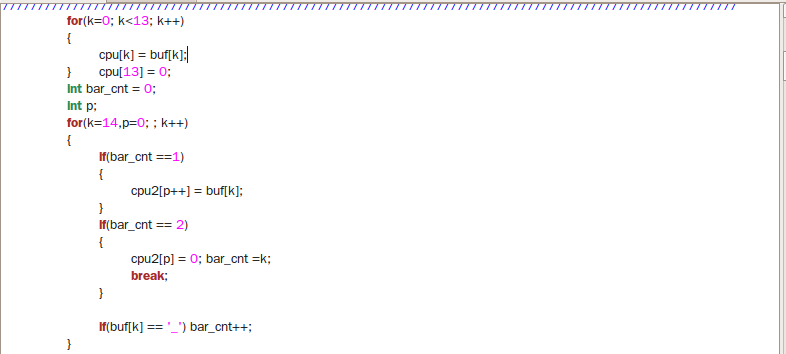
기타장비 : 시스템 전력 프로파일링 장비 - yokogawa WT210, 잘만 MFC3, 랩뷰 소프트웨어

3-5. 코드의 구조 설명

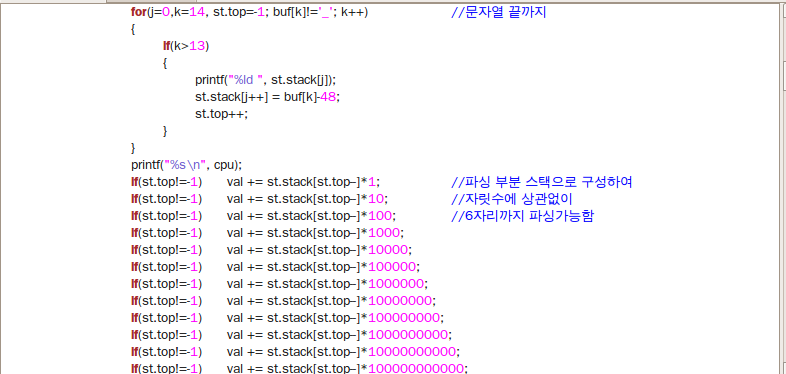
- 개발부분에 대한 기능 및 설명

# 서버마크 관리자 부분

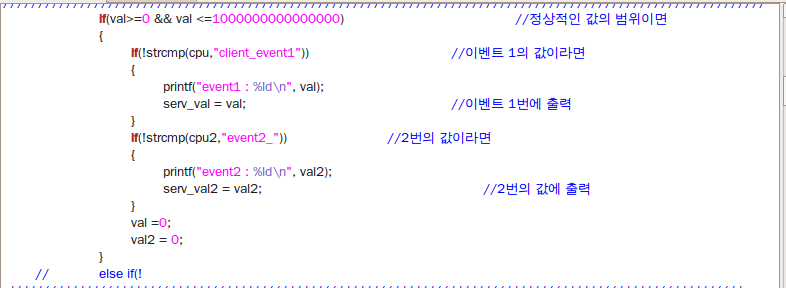
멀티 쓰레드로 구현하여 크게 background server 부와 GUI Interface 부로 나누어진다. background server부의 하위 메소드는 기본적인 서버 구현시 메소드 들과 클라이언트에서 받은 측정 값을 파싱하는 부분으로 이루어져 있다. GUI부는 버튼과 프레임 등의 widget 부분과 그래프등 draw 를 담당하는 on\_expose\_event 부분, 글씨 출력을 담당하는 text 부로 나뉘어 지며 on\_expose\_event 부에서는 Cairo를 이용하여 그래프를 그려준다. text부에서는 그래프 옵션 화면의 글씨 출력을 담당하며 역시 cairo 라이브러리를 이용하여 출력하여 준다. 아래는 주요 소스 구문과 각 구문에 대한 설명이다.



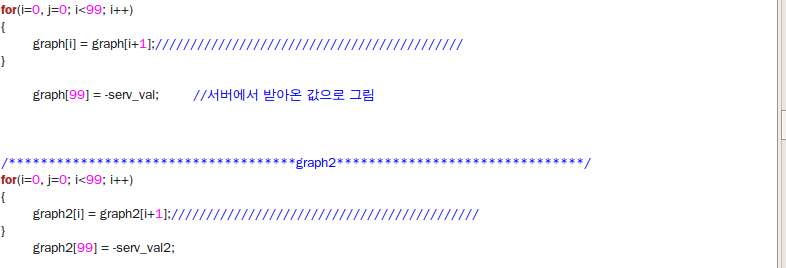
정보를 받아와 파싱 부분을 메시지 구분 기호인를 검색해 내는 부분으로 첫 부분과 값 뒤의 ‘\_’ 로 구분 하기 위한 for문 으로 구성 되어있다.



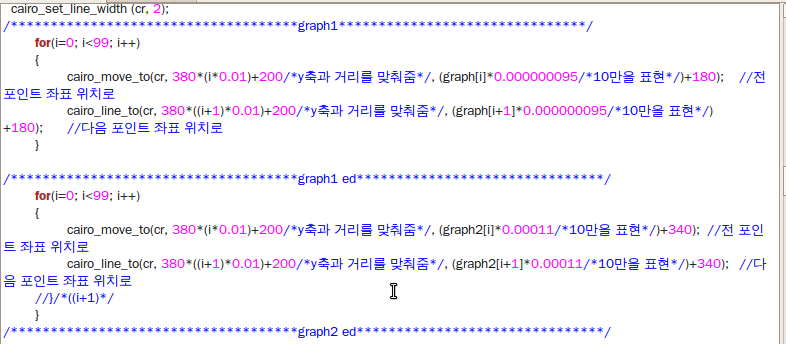
드라이버로부터 수신하는 성능 이벤트 값의 파싱 부분으로 이벤트 값의 자릿수의 변화폭이 크기 때문에 스택을 활용하여 첫 번째 자리부터 계산하고 변수 val2에 저장하게 된다.



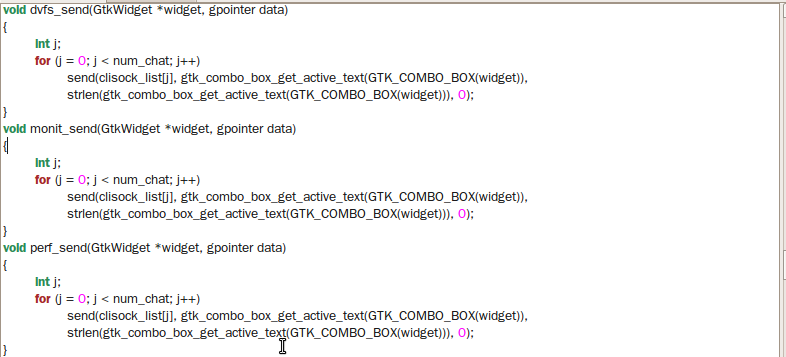
파싱된 메시지 값을 분석하는 부분으로 분기문을 이용하여 각 메시지를 구별하며 수신된 구문을 변수 val, val2에 저장하고 유효범위내의 값일 경우 그래프에 출력하게 된다. 그래프 출력 후에 변수 val, val2는 초기화된다.



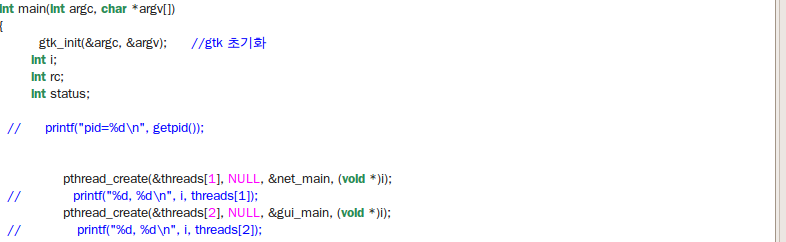
수신한 성능 이벤트 값을 그래프 배열에 넣어주는 구문으로 별도의 그래프 드로잉 라이브러리를 사용하지 않고 단순하게 구현하였다. 그래프 x축 크기(100개) 만큼의 변수 공간을 할당하고, 각각의 값을 저장하여 그래프를 출력한다.



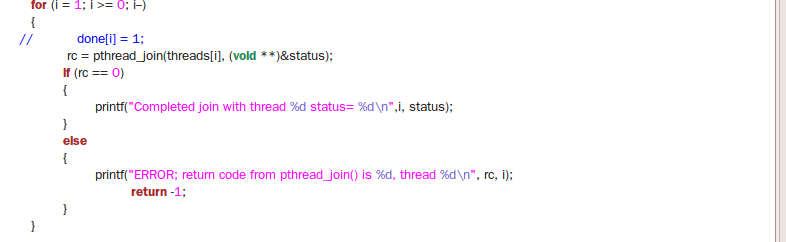
실질적으로 그래프를 그려주는 구문으로 그래프 배열의 값을 이용하여 그래프를 출력한다. cairo 라이브러리를 호출하여 출력하며 graph1 부분에서 위쪽 그래프를, graph2 부분에서 2번 그래프를 출력하게 된다.



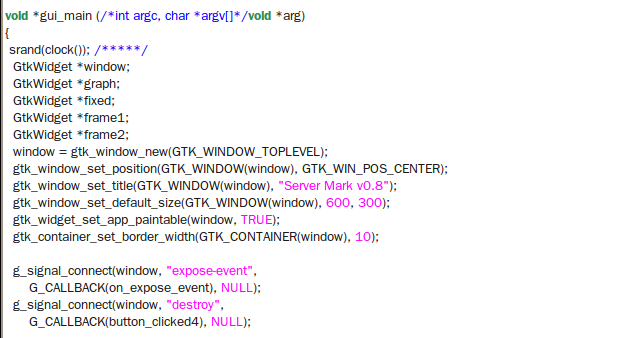
드라이버 설정 부분에서 값을 변경 하였을 때 호출되는 함수들로, 변경 값을 보내는 역할을 하게 된다. 이 함수들이 호출될 경우 net\_main 의 send 함수를 호출하여 해당 옵션 값을 송신하게 된다.



메인 함수 부분으로 각각 GUI 파트와 서버파트의 스레드를 생성한다.



두 개의 스레드를 연결 시켜주는 구문으로 실패할 경우 -1을 리턴한다.



gui\_main 함수로서 랜덤 클럭을 초기화 하고 메인 프레임을 구성할 GTK 위젯들을 생성하고 신호를 연결한다.

# 개발 범위

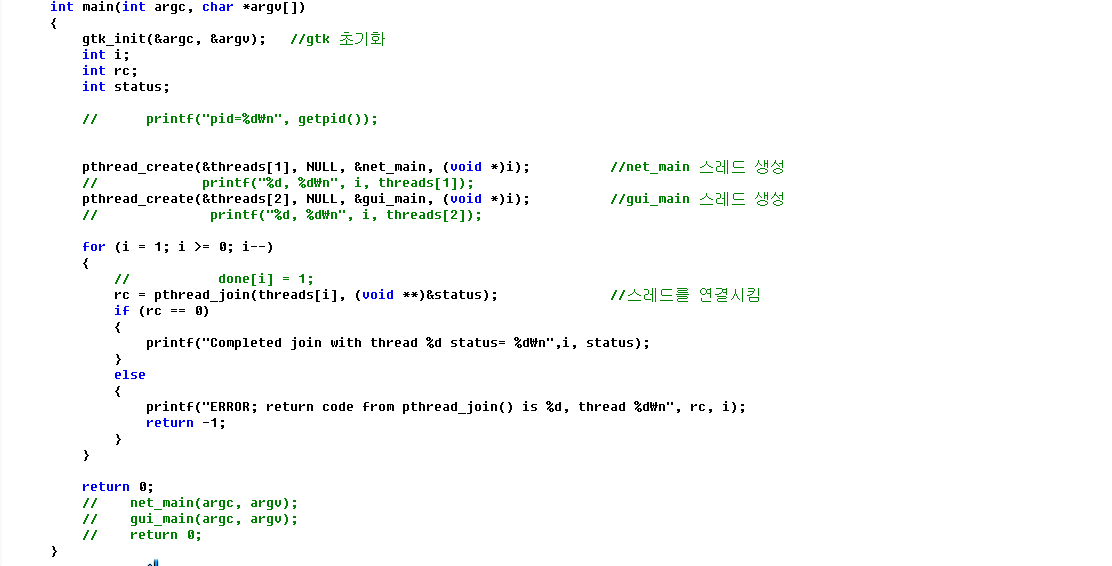
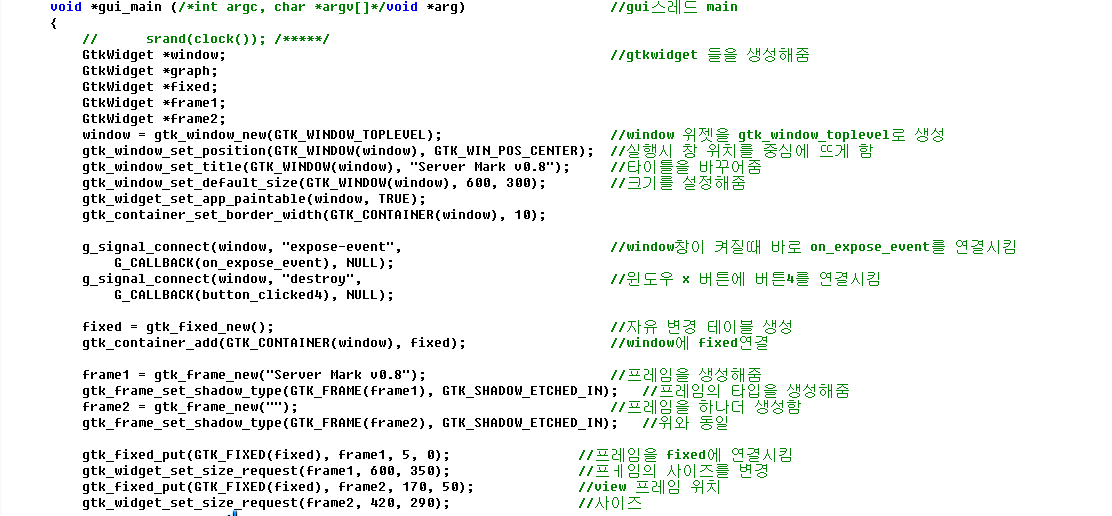
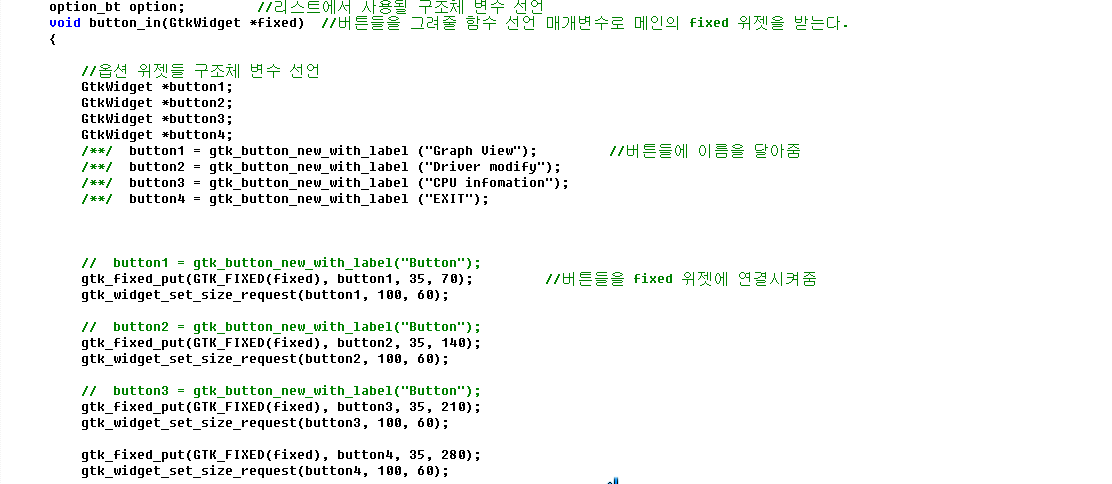
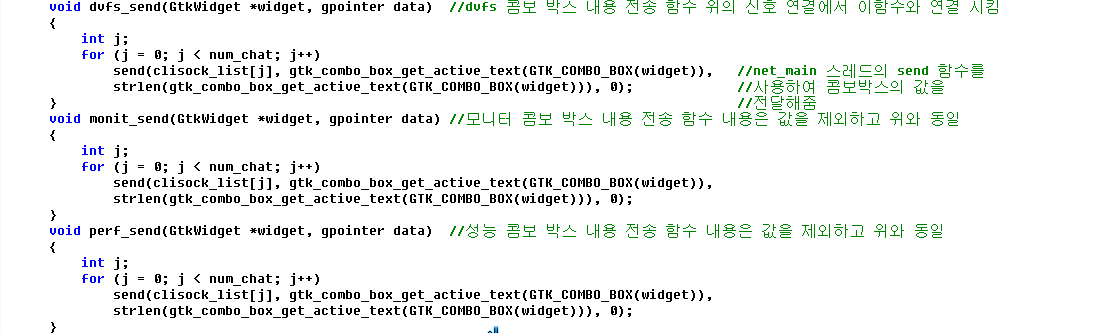
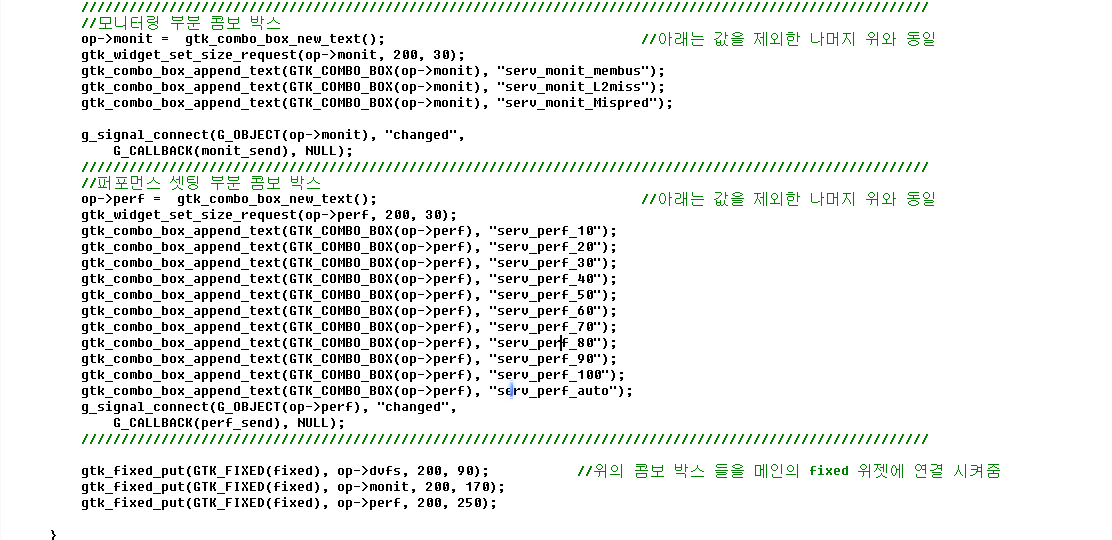
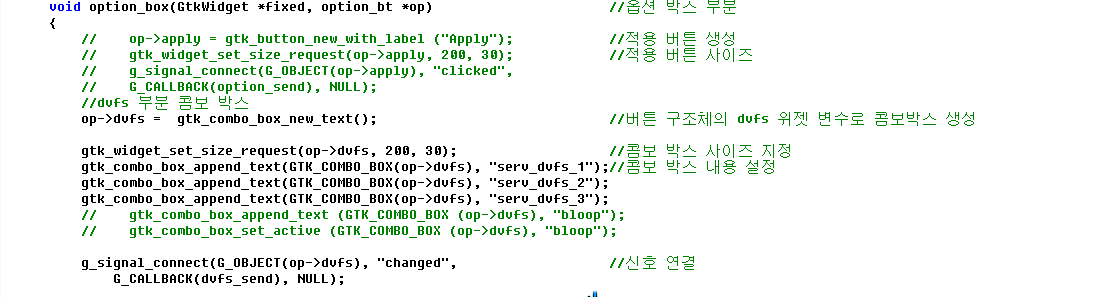
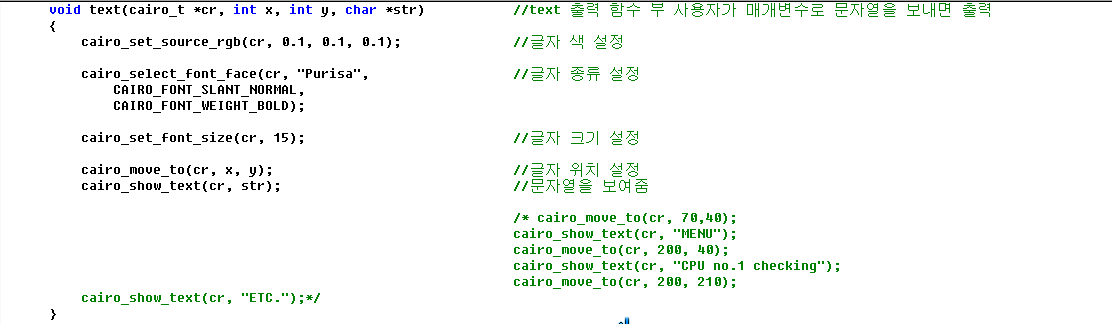
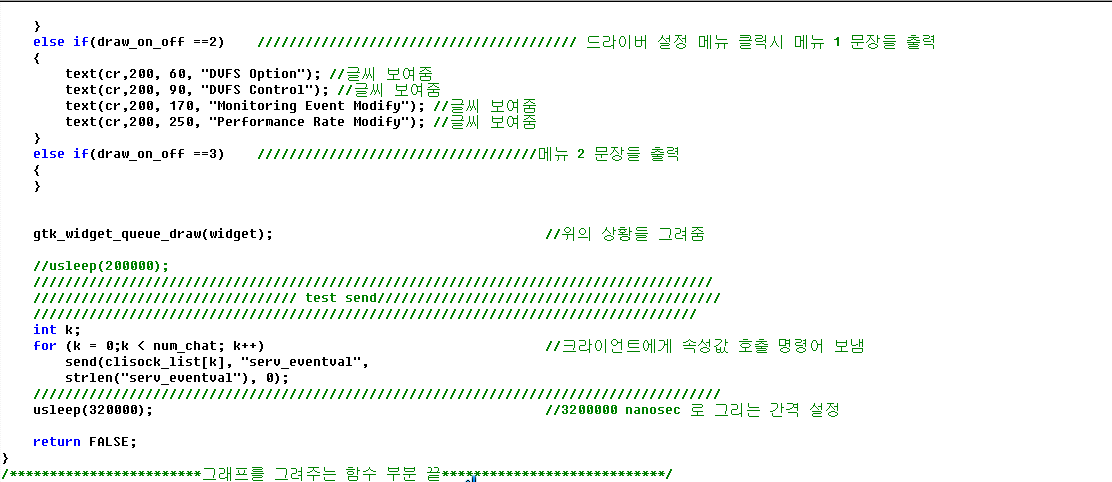
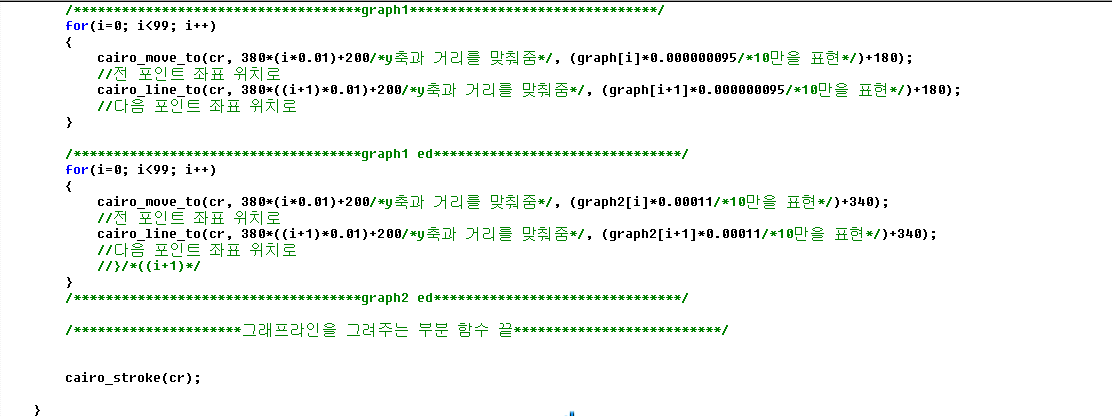
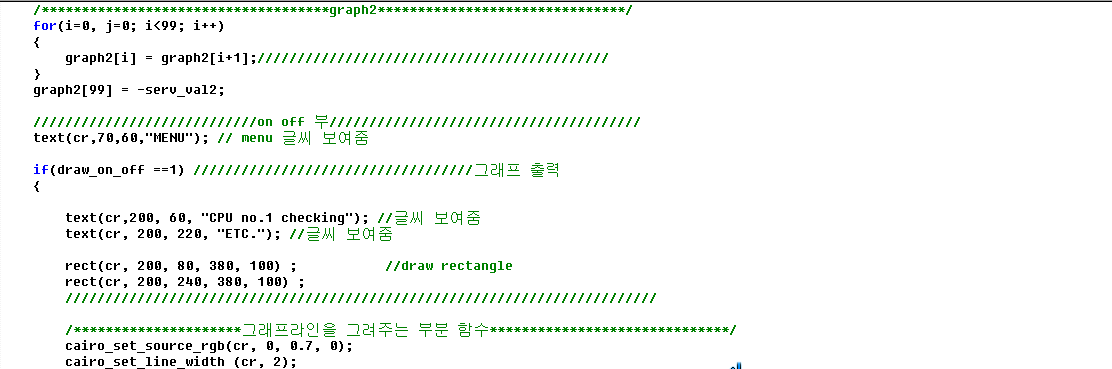
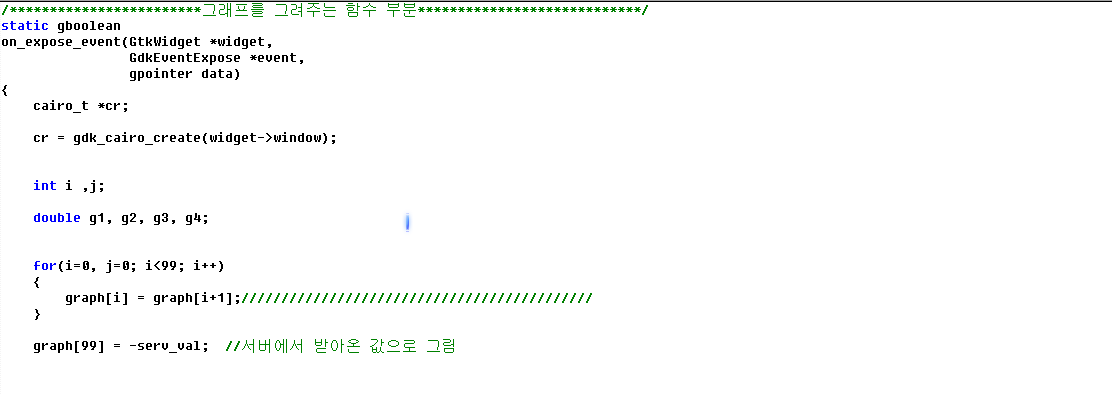
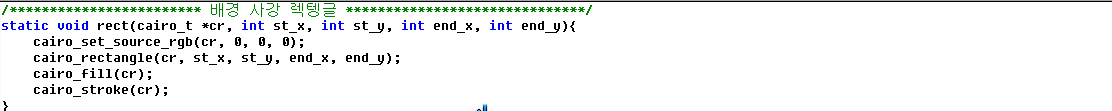
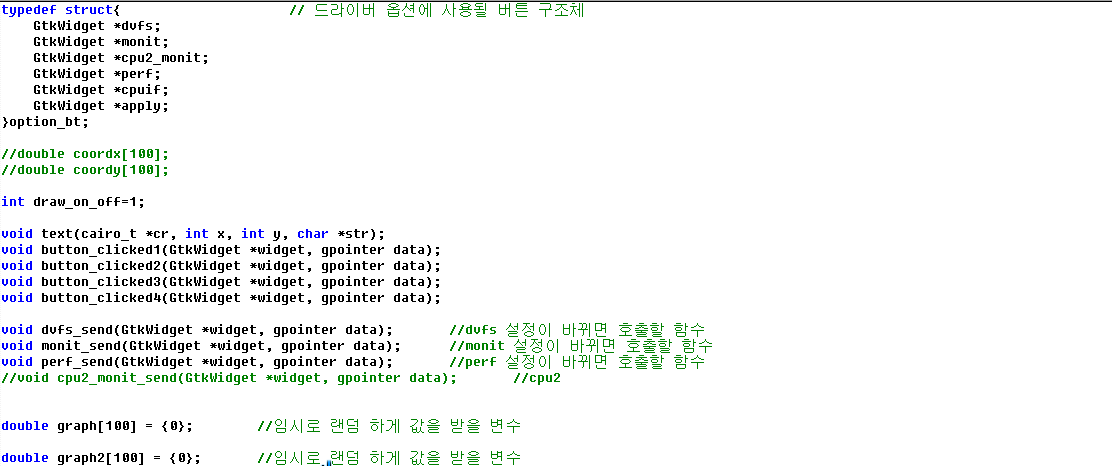
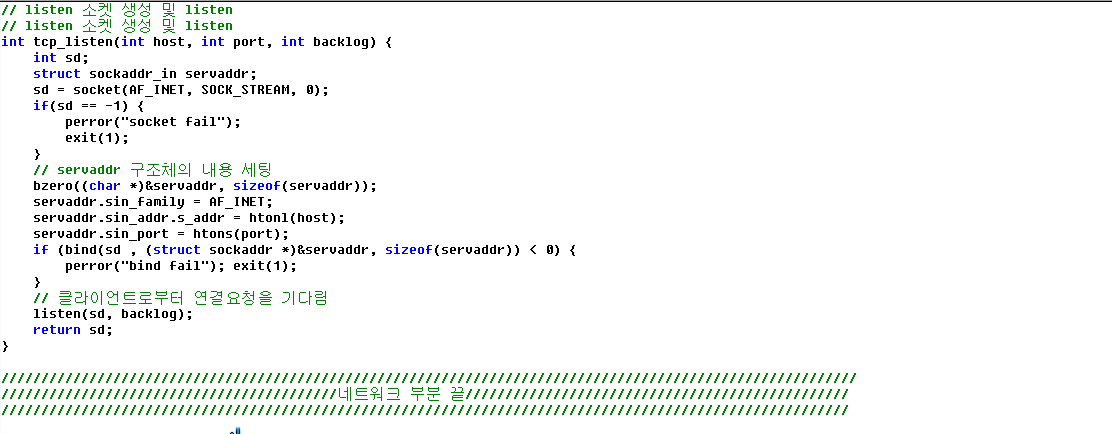
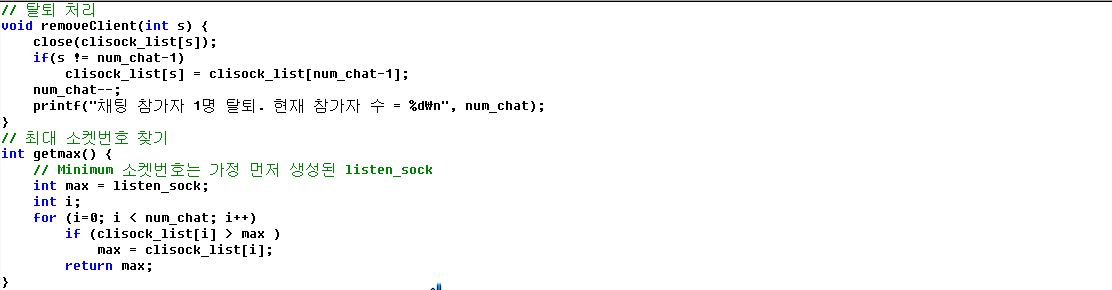
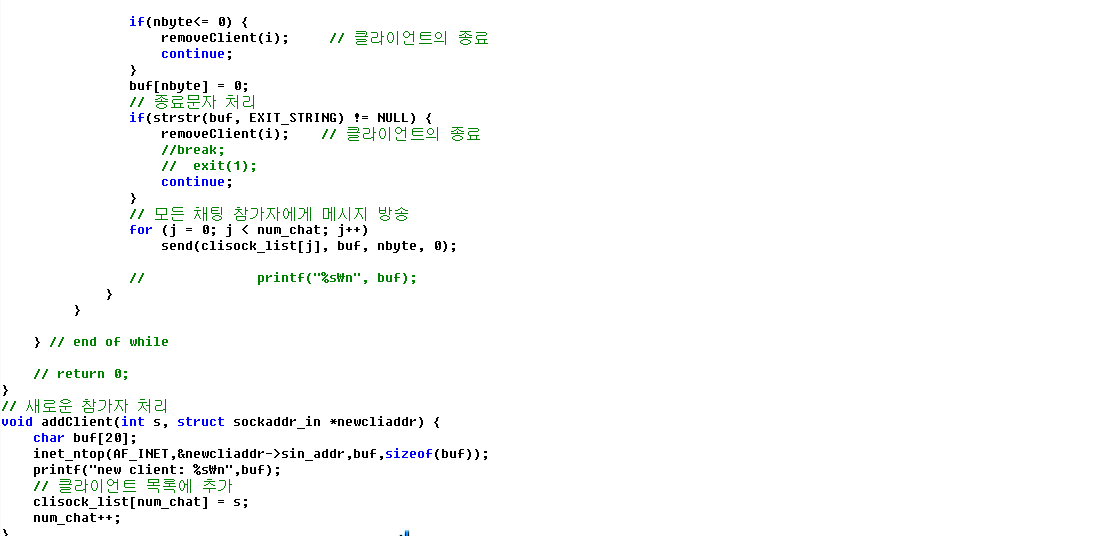
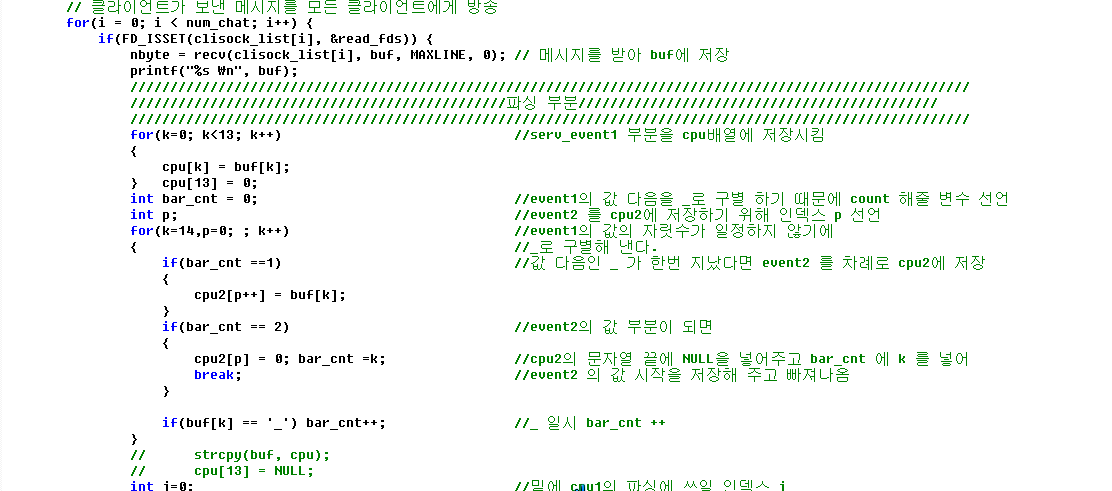
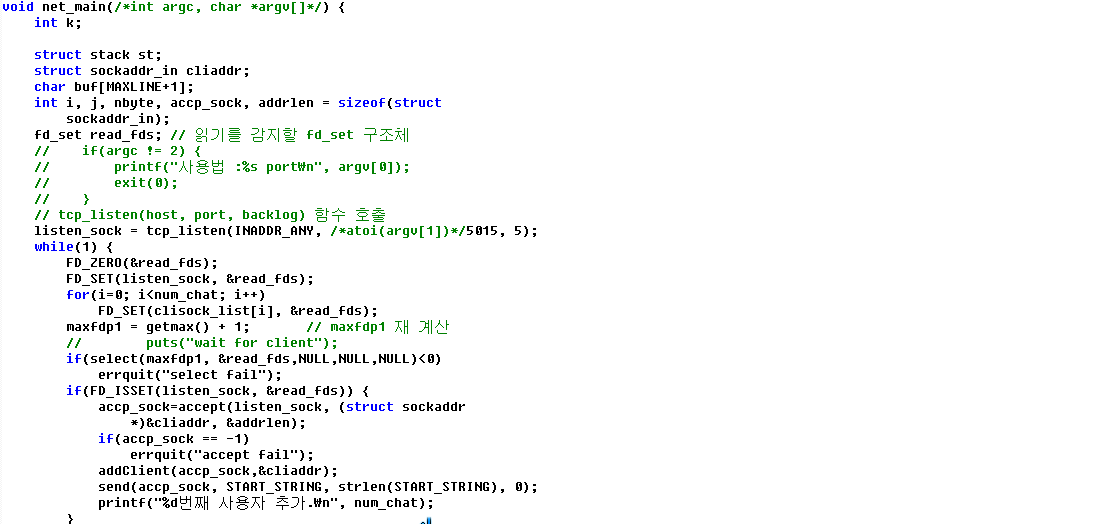
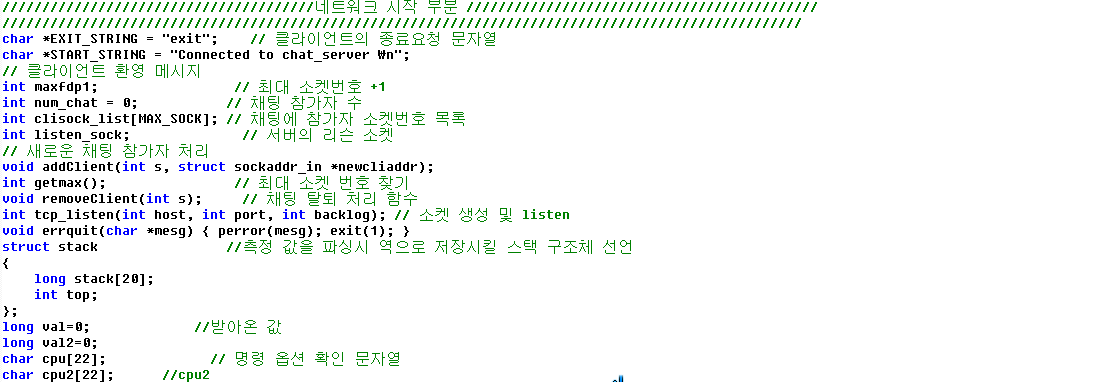
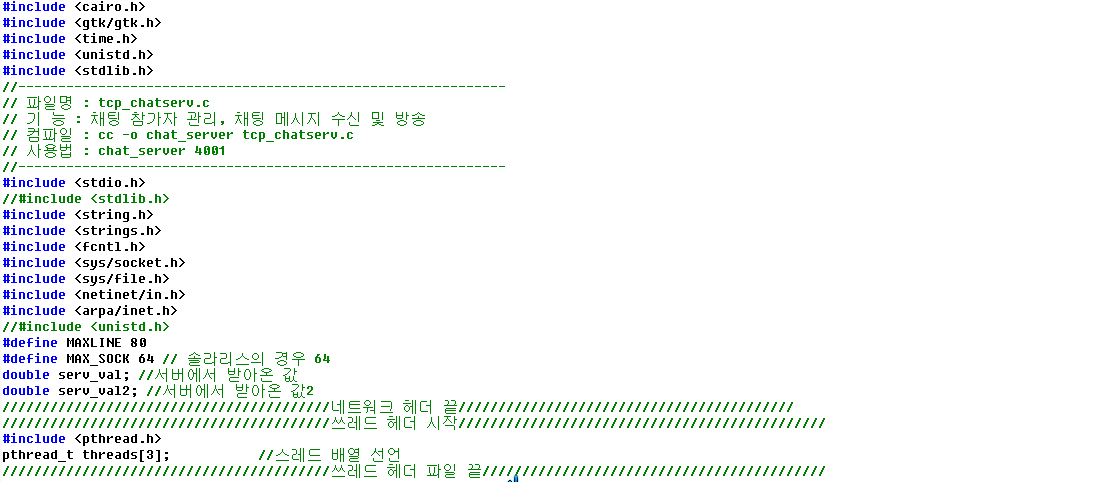
● 금번 개발 범위

서버마크 v0.9 의 개발범위는 각 서버에 설치된 에이전트에서 측정한 성능 이벤트를 획득하여 원격의 관리자 프로그램에서 시스템 사용정도를 분석하고 이에 따라 수동으로 전력 정책을 결정하거나, 자동으로 DVFS 기법을 적용함으로서 전력 사용량을 감소시킨다. 또한 사용자가 관심 있어 하는 하드웨어 성능 이벤트들을 모니터링 할 수 있는 기능을 제공한다.

● 향후 개발 범위

이번 프로그램 개발에서는 시스템 사용정도를 모니터링하여 DVFS를 적용함으로서 전력 소모량을 감소시킬 수 있다는 것을 증명한 것이 가장 큰 기여라고 할 수 있다. 하지만 이 프로그램의 가장 큰 제한점은 실제로 하드웨어 성능 이벤트를 설정하고 DVFS를 적용하는 부분이 플랫폼에 상당히 의존된다는 것이다. 현재는 인텔 코어 아키텍쳐만을 대상으로 하였지만 추후 다양한 아키텍쳐를 동적으로 탐색하고 적합한 작업을 수행할 수 있는 부분을 추가 구현할 예정이다. 또한 관리자 프로그램에서 명령을 송신하였을 때 실제로 그 명령이 적절히 수행되었는지 검증할 수 있는 부분을 추가 구현하고, 발생할 수 있는 다양한 오류에 대한 예외처리 구문을 추가할 예정이다.

# 전체 소스 내용



# 기타

- 업데이트 계획, 특화된 개발기술, 특기사항 등의 설명

● 업데이트 계획

2009. 11.31 데이터 정리 및 홈페이지 구축

- 실질적으로 소스를 공개하고, 다양한 포럼 활동을 시작

2009. 12.31 Ver 1.0 업데이트

- 에러 처리 부분 구현, GUI 메뉴 최적화

2010. ~

- 현재 지원하는 플랫폼 외에 추가 플랫폼 지원

- 공통 플랫폼의 프로파일링 데이터 수집 서버 구현

● 특화된 개발 기술, 구현물 만의 장점

- 낮은 워크로드로 프로세서의 시스템의 사용빈도를 모니터링

- 간편한 인터페이스로 동작하며, 자동 전력관리 수행 가능

- 통합 제어 컴퓨터에서 다수의 서버의 전력관리를 수행함으로서 효율성 극대화

● 참고 사이트, 문헌

1. Cairo is implemented as a library written in the C programming language   
   <http://www.cairographics.org/>
2. The GIMP Toolkit. http://www.gtk.org/