

# 임베디드시스템및실습 과제2 보고서

-Redis의 기능과 원리-

컴퓨터공학과 사재현

컴퓨터공학과 김태형

컴퓨터공학과 심홍재

## 1. redis-server

### 1.1. aeMain()을 중심으로 레디스의 전반적인 작동 원리 분석

ae.c:aeMain()은 Redis의 이벤트 루프를 구현하는 부분이다. 이벤트 루프란 서버가 클라이언트로부터 메시지를 요청받고 이에 대한 응답을 반환하기 위해 계속해서 이벤트를 대기하는 상태를 유지, 반복하는 프로그래밍 방법론이다. 이 소스코드는 파일 디스크립터(유닉스 계열 운영체제에서 프로세스가 특정 파일에 접근할 때 사용하는 추상 값)의 변화나 타이머 이벤트 등을 감지하고 처리하는 내용을 담고 있다. 다음은 코드의 주요 함수와 그 기능에 대한 간략한 설명이다

#### a. 이벤트 루프 초기화 및 생성

- aeCreateEventLoop: 이벤트 루프를 초기화하고 생성. 주어진 크기로 이벤트 구조체를 할당하고 초기화.

#### b. 이벤트 루프 크기 및 속성 변경

- aeGetSetSize: 현재 이벤트 루프의 크기를 반환.
- aeSetDontWait: 이벤트 루프의 대기 타임아웃을 변경.

#### c. 파일 이벤트 처리

- aeCreateFileEvent: 파일 이벤트를 생성하고 등록.
- aeDeleteFileEvent: 파일 이벤트를 제거.
- aeGetFileClientData: 파일 이벤트와 관련된 클라이언트 데이터를 반환.
- aeGetFileEvents: 파일 이벤트의 현재 상태를 반환.

d. 타이머 이벤트 처리

- aeCreateTimeEvent: 타이머 이벤트를 생성하고 등록.
- aeDeleteTimeEvent: 타이머 이벤트를 제거.
- usUntilEarliestTimer: 가장 빠르게 발생할 타이머 이벤트까지 남은 마이크로초를 반환.
- processTimeEvents: 현재 시간에 발생해야 할 타이머 이벤트를 처리.

e. 이벤트 처리 메인 루프

- aeProcessEvents: 주어진 이벤트 플래그에 따라 파일 및 타이머 이벤트를 처리.
- aeWait: 주어진 파일 디스크립터가 지정된 이벤트 상태가 될 때까지 대기.
- aeMain: 주 이벤트 루프 메서드로, 이벤트 처리를 계속 반복.

f. 이벤트 루프 실행

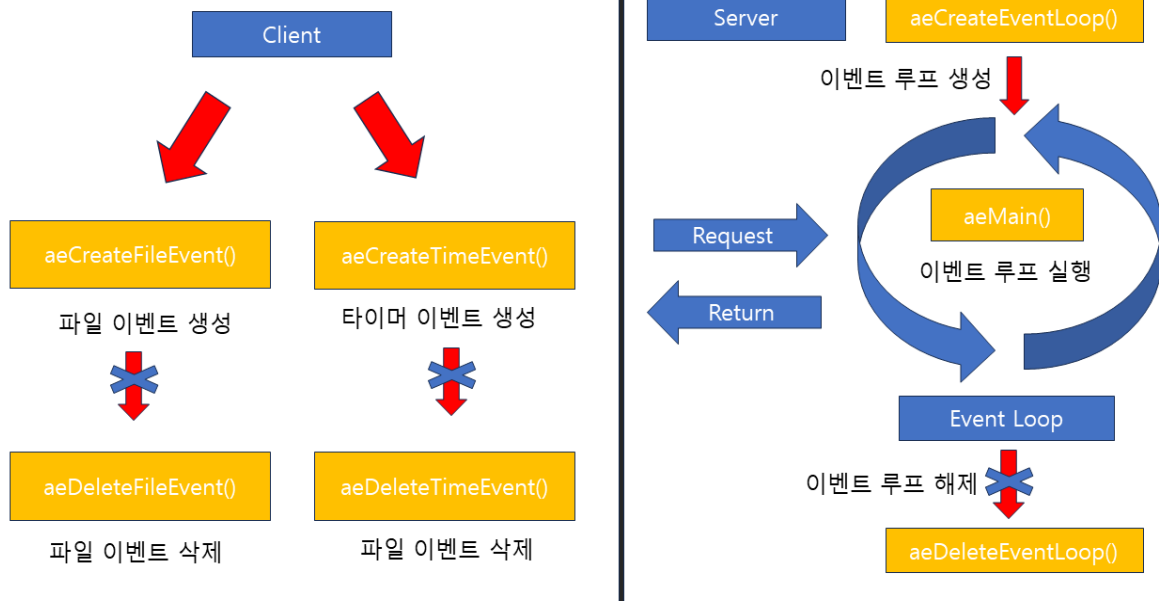
- aeMain: 주 이벤트 루프를 실행. 루프를 멈추려면 aeStop을 호출.

g. 이벤트 루프 종료 및 정리

- aeDeleteEventLoop: 이벤트 루프와 관련된 자원을 해제.

정리하면, 레디스 서버는 이벤트의 수신을 계속 대기하고 수신 받은 이벤트의 응답을 클라이언트로 반환하는 무한반복 상태를 유지하며 이를 이벤트 루프라 한다. 레디스 이벤트 루프는 aeCreateEventLoop 함수 호출로 실행되고 파일 이벤트는 aeCreateEventFile, 타이머 이벤트는 aeCreateTime으로 생성 및 등록된다. 이벤트 루프의 종료는 aeDeleteEventLoop함수의 호출로 이루어진다. 아래는 레디스 서버의 전반적인 동작 과정을 나타낸 그림이다.

### 1-1. 레디스의 전반적인 작동 원리



<그림 1 - 레디스 이벤트 루프의 전반적인 동작 원리>

### 1-2. 레디스 SET 명령어 동작 원리 분석

SET 명령어는 레디스에서 사용되는 기본적인 명령어 중 하나로, 키-값 쌍을 저장하는 데 사용된다. 이 명령어는 주어진 키에 값을 할당하여 레디스의 데이터 저장 및 관리를 위한 핵심적인 기능 중 하나이다. SET 명령어를 통해 값을 저장하는 과정은 아래의 단계로 구성된다.

#### 1) 클라이언트-서버 연결

클라이언트가 `redis-cli`를 사용하여 서버에 접속한다. TCP/IP 소켓 연결을 통해 클라이언트와 서버 사이의 통신이 이루어진다.

#### 2) SET 명령어 전송

클라이언트가 SET 명령어와 함께 key-value 쌍을 서버에 전송한다.

#### 3) 서버에서의 처리

서버는 받은 SET 명령어를 처리한다. 데이터는 메모리 내의 키-값 구조로 저장된다.

#### 4) 사용되는 자료구조

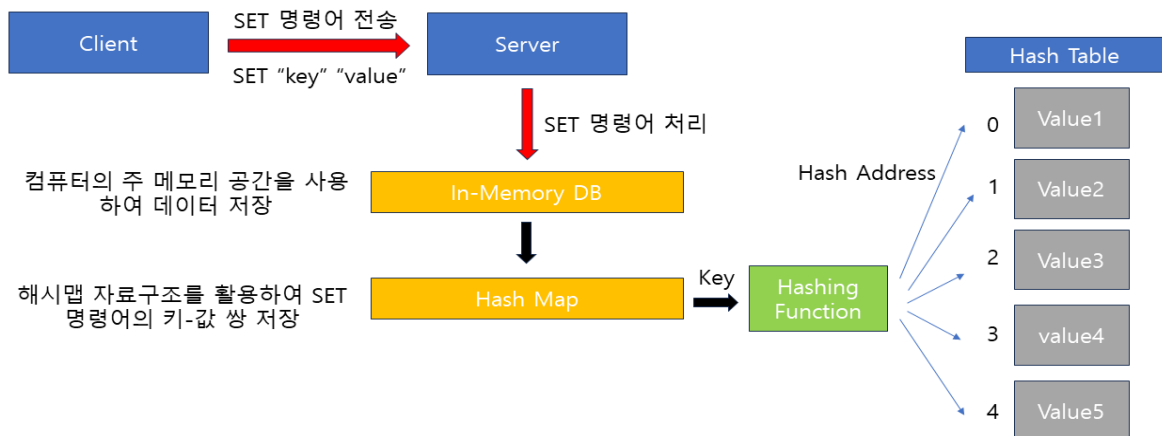
레디스는 간단한 키-값 저장소이지만 내부적으로 여러 자료구조를 사용한다. 주로 해시맵(Hash Map)을 사용하여 키와 값을 빠르게 검색하고 접근할 수 있도록 한다.

여기서 해시맵이란, 해싱과 맵을 결합한 것으로 맵은 키-값의 쌍을 이루는 자료구조, 해싱은 키 값에 산술적인 연산을 적용해 찾고자 하는 자료에 접근하는 방법을 뜻한다. 즉, 자료 탐색시 키를 입력 받아 해싱 함수를 통해 해시 주소를 생성하고 이 해시 주소를 맵의 인덱스로 사용하는 탐색 기법이 해시맵인 것이다.

## 5) 값의 저장 형식

값은 일반적으로 문자열, 숫자, 리스트, 해시, 세트, 정렬된 세트 등 다양한 데이터 타입으로 저장될 수 있다. 이러한 데이터는 이진 안전 문자열로 저장되며, 이는 클라이언트가 전송한 데이터 형식을 유지하면서 서버에 저장된다.

### 1-2. SET 명령어 처리 과정



<그림2 – SET 명령어 처리 과정과 사용되는 자료구조>

### 1-2. Call-Graph



<그림3 – SET 명령어 동작과정의 개괄적인 콜 그래프>

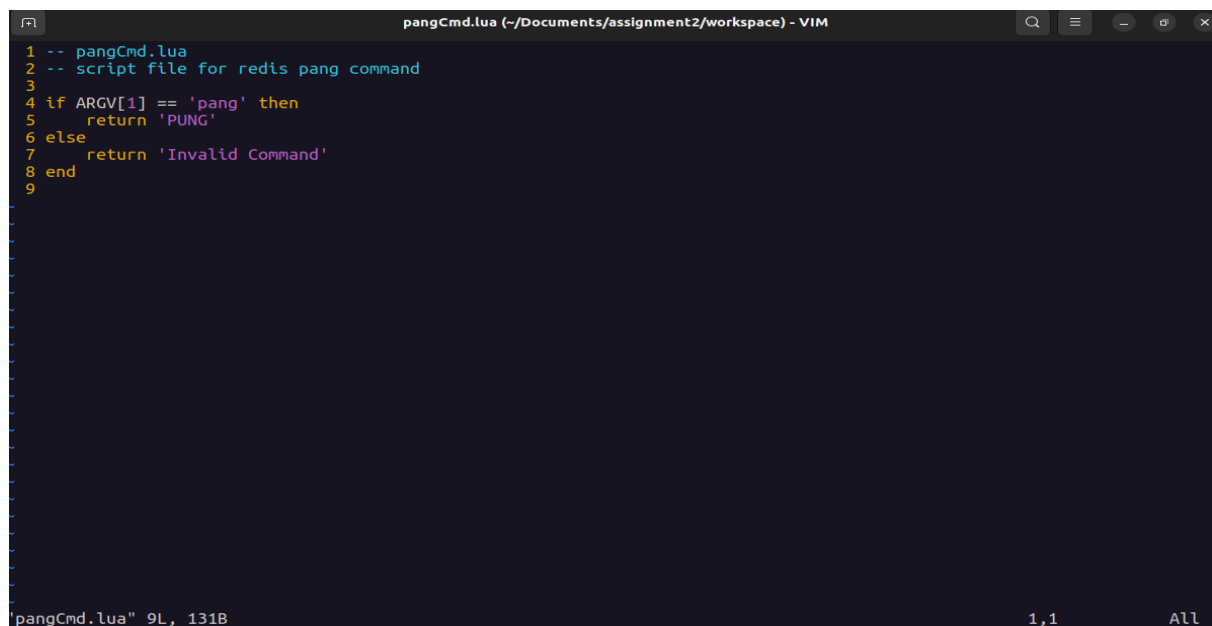
### 1-3. 레디스 pang 명령어 추가

레디스에 직접적으로 명령어를 추가하는 기능은 없다. 그러나 레디스는 EVAL 명령어를 통해 Lua 형식의 스크립트를 실행하는 것이 가능하므로, 이를 이용해 pang을 입력하면 PUNG을 반환하도록 하는 스크립트를 만들어 레디스에서 실행하는 방법으로 기능을 구현해볼 수는 있다. 방법은 다음과 같다.

1) 아래와 같이 pangCmd.lua 스크립트를 만든다.

```
if ARGV[1] == "pang" then
    return "PUNG"
else
    return "Invalid Command"
end
```

스크립트와 함께 받은 명령어 인자가 "pang"인 경우, "PUNG"을 반환하고, 그렇지 않다면 "Invalid Command"를 반환한다.

A screenshot of a VIM editor window titled 'pangCmd.lua (~/.Documents/assignment2/workspace) - VIM'. The editor shows a Lua script with the following content:

```
1 -- pangCmd.lua
2 -- script file for redis pang command
3
4 if ARGV[1] == 'pang' then
5     return 'PUNG'
6 else
7     return 'Invalid Command'
8 end
9
```

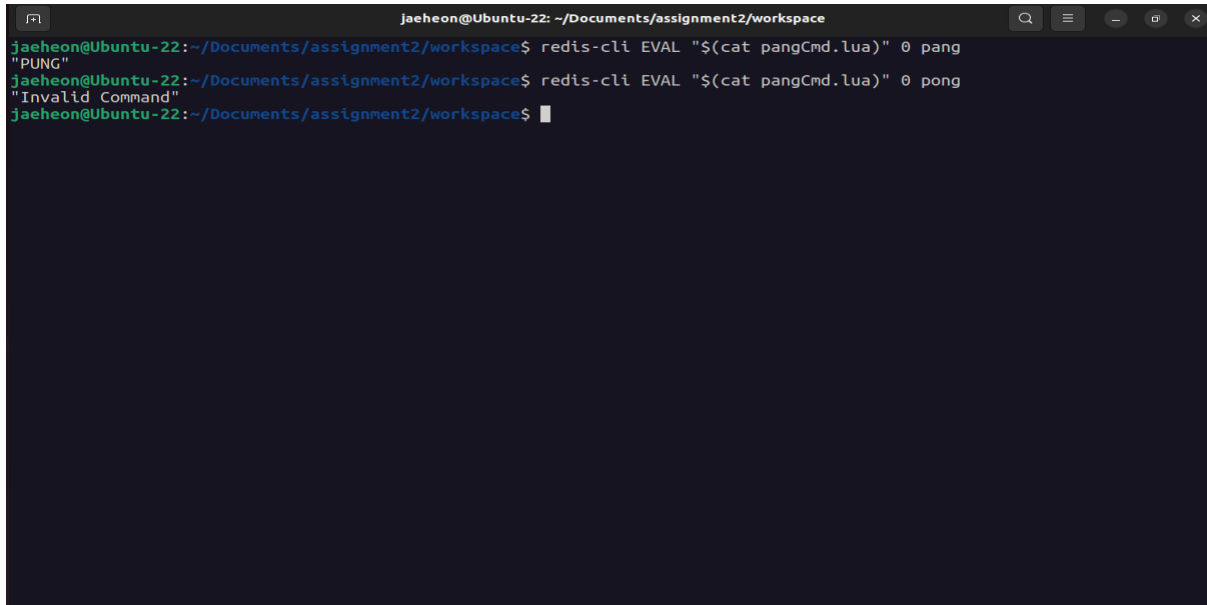
The status bar at the bottom left shows 'pangCmd.lua' 9L, 131B. The status bar at the bottom right shows '1,1' and 'All'.

<그림 4 – pangCmd.lua>

2) 스크립트 파일을 레디스 서버 작업을 진행하는 디렉토리에 위치시킨 후 터미널에 아래와 같은 명령어를 입력한다.

```
$ redis-cli EVAL "$(cat pangCmd.lua)" 0 pang
```

이 명령어는 EVAL 명령어로 스크립트 내용과 함께 pang 문자열을 받아들이며, 스크립트의 내용에 따라 "PUNG"을 출력하게 된다. 0은 스크립트 실행 시 특정 키 값이 필요하지 않음을 의미한다.

A terminal window titled 'jaeheon@Ubuntu-22: ~/Documents/assignment2/workspace'. It shows three lines of Redis CLI commands and their outputs. The first command is 'redis-cli EVAL "\$(cat pangCmd.lua)" 0 pang' which outputs '"PUNG"'. The second command is 'redis-cli EVAL "\$(cat pangCmd.lua)" 0 pong' which outputs '"Invalid Command"'. The third line shows the prompt 'jaeheon@Ubuntu-22:~/Documents/assignment2/workspace\$' with a cursor.

```
jaeheon@Ubuntu-22:~/Documents/assignment2/workspace$ redis-cli EVAL "$(cat pangCmd.lua)" 0 pang
"PUNG"
jaeheon@Ubuntu-22:~/Documents/assignment2/workspace$ redis-cli EVAL "$(cat pangCmd.lua)" 0 pong
"Invalid Command"
jaeheon@Ubuntu-22:~/Documents/assignment2/workspace$
```

<그림 5 - 명령어 실행 결과>

## 2. In-Memory DB

### 2-1. 레디스 데이터베이스의 자료구조 분석

아래는 레디스 데이터베이스에서 자료를 저장하는데 사용되는 자료구조와 그에 대한 간단한 설명이다.

#### 1) String (문자열)

문자열 데이터를 저장한다.

예: SET key value, GET key 등

#### 2) Hash (해시):

필드-값 쌍을 저장하는 데이터 구조이다.

예: HSET myhash field1 "value1", HGET myhash field1 등

#### 3) List (리스트):

순서가 있는 문자열 엘리먼트의 집합을 저장한다.

예: LPUSH mylist "value1", LRANGE mylist 0 -1 등

#### 4) Set (집합)

중복이 허용되지 않는 문자열 엘리먼트의 집합을 저장한다.

예: SADD myset "value1", SMEMBERS myset 등

#### 5) Sorted Set (정렬된 집합):

중복이 허용되지 않는 문자열 엘리먼트와 각 엘리먼트에 대한 스코어(정렬 기준)를 저장한다.

예: ZADD myzset 1 "value1", ZRANGE myzset 0 -1 WITHSCORES 등

#### 6) Bitmaps (비트맵)

각 비트가 특정 오프셋에 대응되는 비트 배열을 저장한다. 비트 값을 통해 간단한 비트 연산이 가능하다.

예: SETBIT mybitmap 0 1, GETBIT mybitmap 0 등

#### 7) HyperLogLog

고유한 값을 추정하는 데 사용되는 확률적 자료 구조이다.

예: PFADD myhyperloglog element1, PFCOUNT myhyperloglog 등

#### 8) Geospatial Index (공간 인덱스)

지리적 위치 정보를 저장하고 검색하는데 사용된다.

예: GEOADD mygeo 13.361389 38.115556 "Palermo", GEODIST mygeo city1 city2 등

### 2-2. String, List 자료형 분석

#### 1) String(문자열)

레디스의 문자열은 바이너리 데이터로 구성된 안전한 문자열로 키와 연관된 값을 저장한다. 내부적으로는 C 문자열과 유사하게 문자 배열로 표현된다.

문자열은 레디스의 SDS(Simple Dynamic Strings)라 불리는 특별한 문자열 표현 방식을 사용한다. SDS는 가변 길이 문자열로, 문자열 길이와 할당된 버퍼의 크기를 함께 저장하여 문자열의 연산을 빠르게 수행할 수 있도록 한다. SDS는 문자열의 길이 증가, 감소에 따라 메모리를 동적으로 할당하거나 해제하여 메모리 효율을 유지한다.

- 명령어 예시:

```
SET mykey "Hello"
```

```
GET mykey
```

## 2) List

레디스의 리스트는 연결 리스트와 유사한 형태로 여러 값을 저장한다. 리스트의 노드는 각각의 값과 다음 노드를 가리키는 포인터를 포함한다.

각 리스트는 head와 tail 포인터를 갖고 있어서 이를 이용해 양쪽 끝에서의 노드 삽입 및 삭제가 이루어진다. 또한 이중 연결 리스트로 구현되어 있어 양방향 순회가 가능하다.

- 명령어 예시:

LPUSH mylist "World" (mylist에 "World"를 추가)

LPOP mylist (mylist의 헤드에서 요소를 반환하고 제거)

LLEN mylist (mylist의 길이 반환)

## 2-3. 레디스의 Data Type에 이진 탐색 트리 추가

2-3문제는 해결하지 못했습니다.

[깃허브 레포지터리]

<https://github.com/Zaixian5/LinuxistAssignment2>