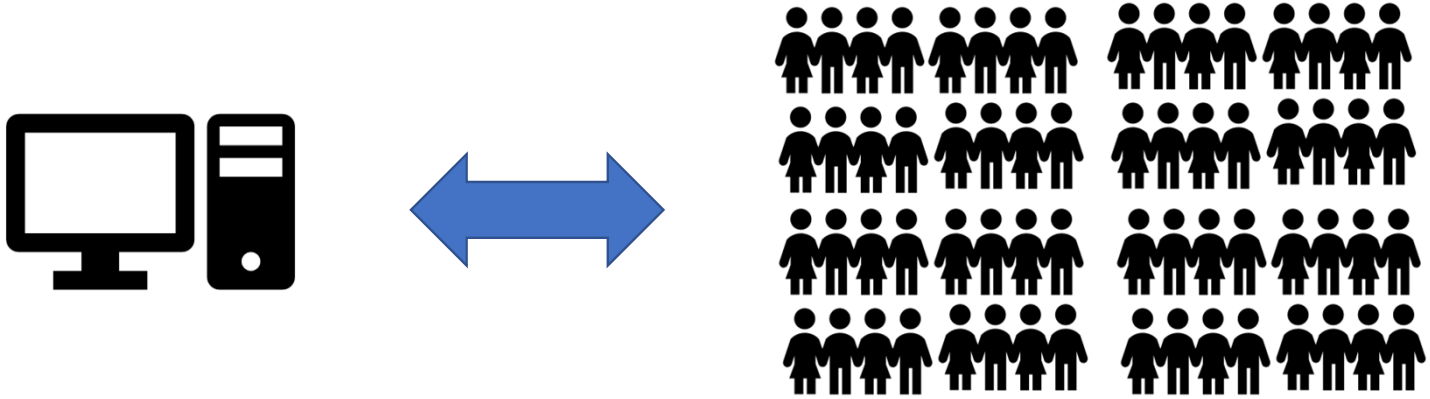
EPOLL in Redis

석사과정 성한승

Epoll?



Epoll functions

- Int epoll_create(int size);
- Int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);
- Int epoll_wait(int epfd, int struct epoll_event *event, int timeout);

Epoll functions

- epoll_create : size 만큼 epoll 객체 공간을 생성

- 형태

```
int epoll_create(int size);
//반환 값 : 실패 시 -1, 일반적으로 epoll_fd의 값을 리턴
```

- 예시

```
int fd_epoll;
bool is_epoll_init = false;
int EpollInit(int size) {

        if((fd_epoll = epoll_create(size)) > 0)
            is_epoll_init = true;
            return fd_epoll;
 }
```

Epoll functions

- epoll_ctl : 할당해놓은 epoll공간에 fd와 event를 등록, 삭제 및 변경
- 형태

```
int epoll_ctl(int epoll_fd,                  //epoll_fd
              int operate_enum,              //어떤 변경을 할지 결정하는 enum값
              int enroll_fd,                 //등록할 fd
              struct epoll_event* event      //관찰 대상의 관찰 이벤트 유형
              );
//반환 값 : 실패 시 -1, 성공시 0
```

Operate_enum

EPOLL_CTL_ADD
EPOLL_CTL_DEL
EPOLL_CTL_MOD

epoll_event 구조체

```
struct epoll_event
{
  __uint32_t events;
  epoll_data_t data;
}

Typedef union epoll_data
{
   void* ptr;
   int fd;
   __uint32_t u32;
   __uint64_t u64;
}epoll_data_t;
```

event 종류

```
enum Events
{
   EPOLLIN,          //수신할 데이터가 있다.
   EPOLLOUT,         //데이터 전송 가능하다.
   EPOLLPRI,         //중요한 데이터(OOB)가 발생.
   EPOLLRDHUP,       //연결 종료 or Half-close 발생
   EPOLLERR,         //에러 발생
   EPOLLET,          //엣지 트리거 방식으로 설정
   EPOLLONESHOT, //한번만 이벤트 받음
}
```

Epoll functions

- epoll_ctl : 할당해놓은 epoll공간에 fd와 event를 등록, 삭제 및 변경
- 형태

```
int epoll_ctl(int epoll_fd,                    //epoll_fd
              int operate_enum,                //어떤 변경을 할지 결정하는 enum값
              int enroll_fd,                   //등록할 fd
              struct epoll_event* event        //관찰 대상의 관찰 이벤트 유형
              );
//반환 값 : 실패 시 -1, 성공시 0
```

- 예시

```
int EpollAdd(const int fd) {

              struct epoll_event ev;
              ev.events = EPOLLIN | EPOLLOUT | EPOLLERR;
              ev.data.fd = fd;
              return epoll_ctl(fd_epoll, EPOLL_CTL_ADD, fd, &ev);
}
```

Epoll functions

- epoll_wait : 이벤트가 발생했는지 검사하는 함수

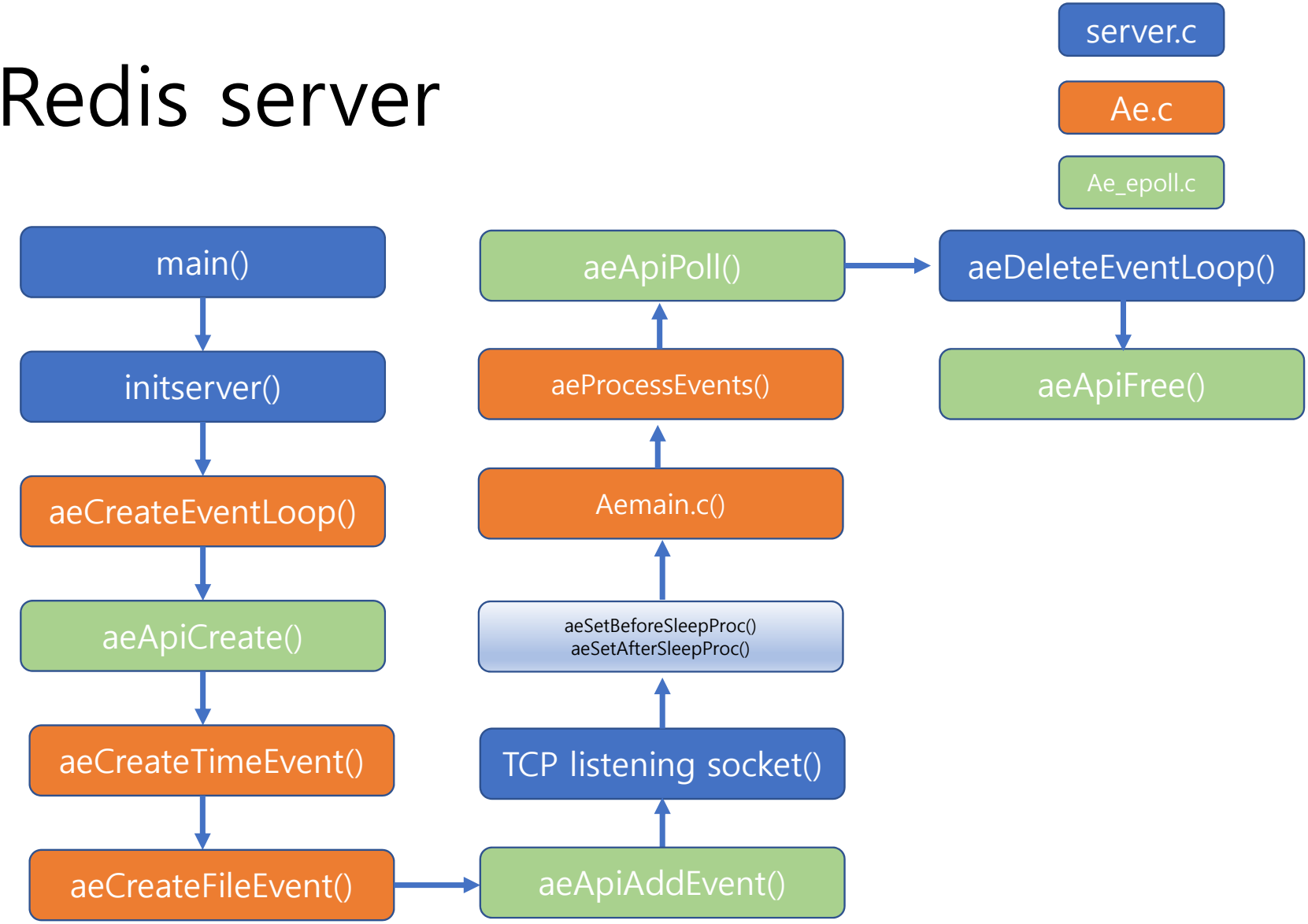
- 형태

```
int epoll_wait( int epoll_fd,                    //epoll_fd
                struct epoll_event* event,       //event 버퍼의 주소
                int maxevents,                   //버퍼에 들어갈 수 있는 구조체 최대 개수
                int timeout                      //select의 timeout과 동일 단위는 1/1000
              );
//성공시 이벤트 발생한 파일 디스크립터 개수 반환, 실패시 -1 반환
```

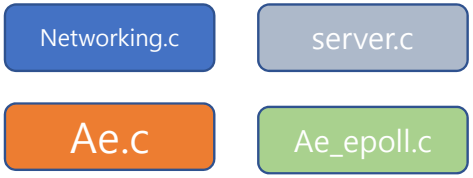
- 예시

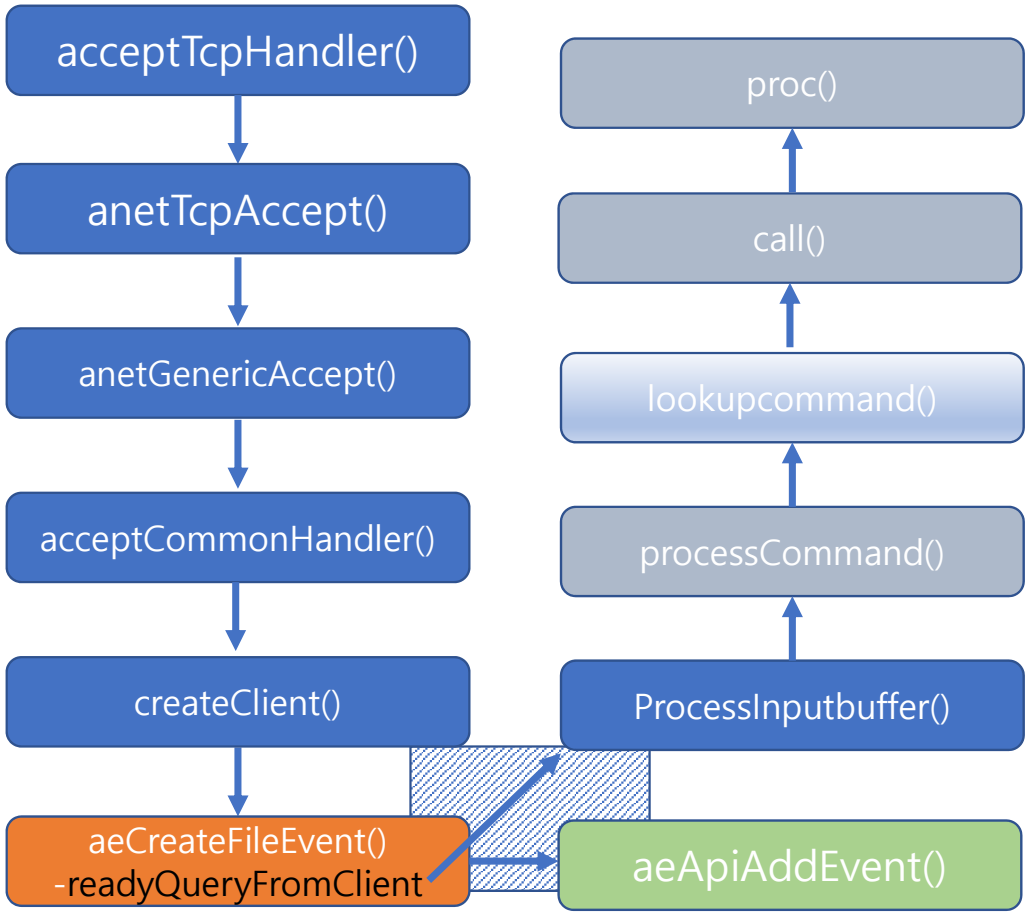
```
While(1){
        event_cnt = epoll_wait(epfd, ep_events, EPOLL_SIZE, -1);

        if(event_cnt==-1){
                puts("error");
                break;
          }

        for(i=0; I < event_cnt; i++){
                . . . . . .
        }
}
```

Redis server

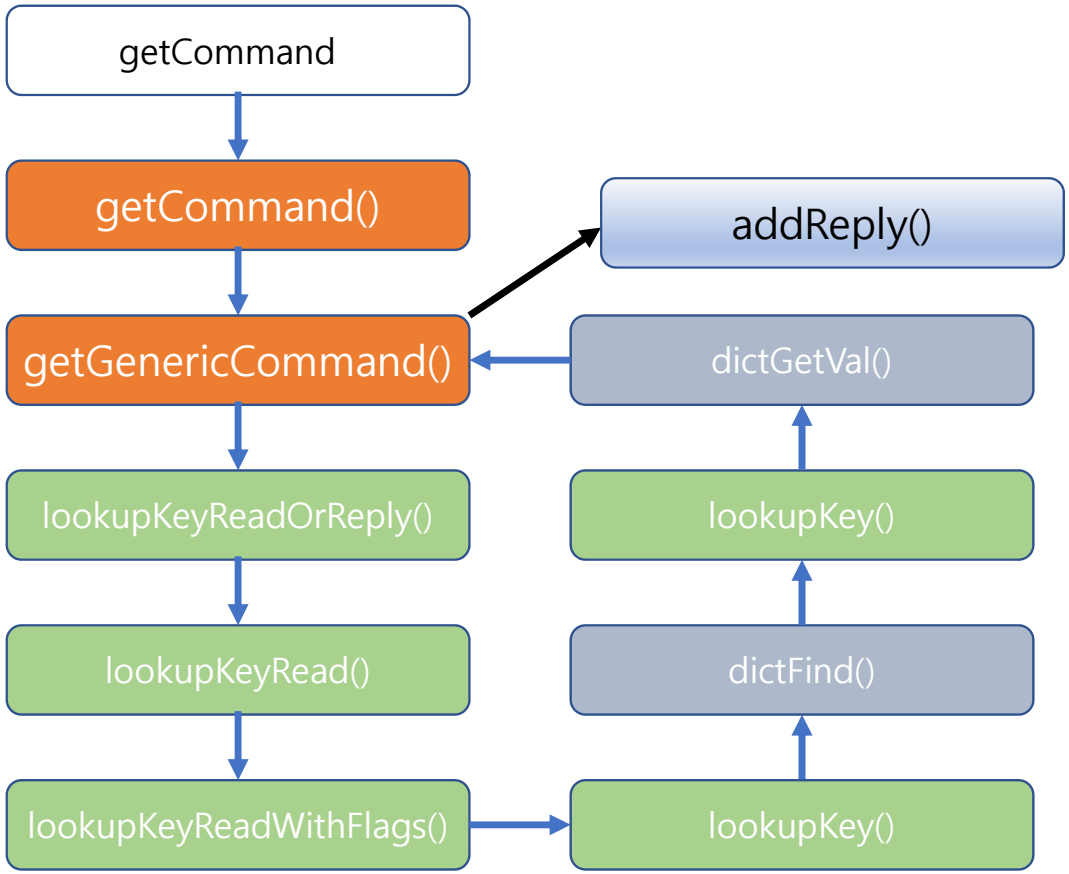


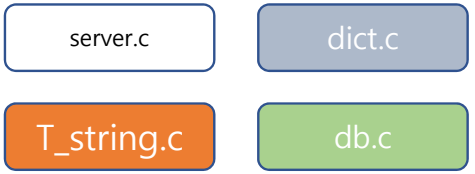
Accept -> call


Networking.c
server.c
Ae.c
Ae_epoll.c

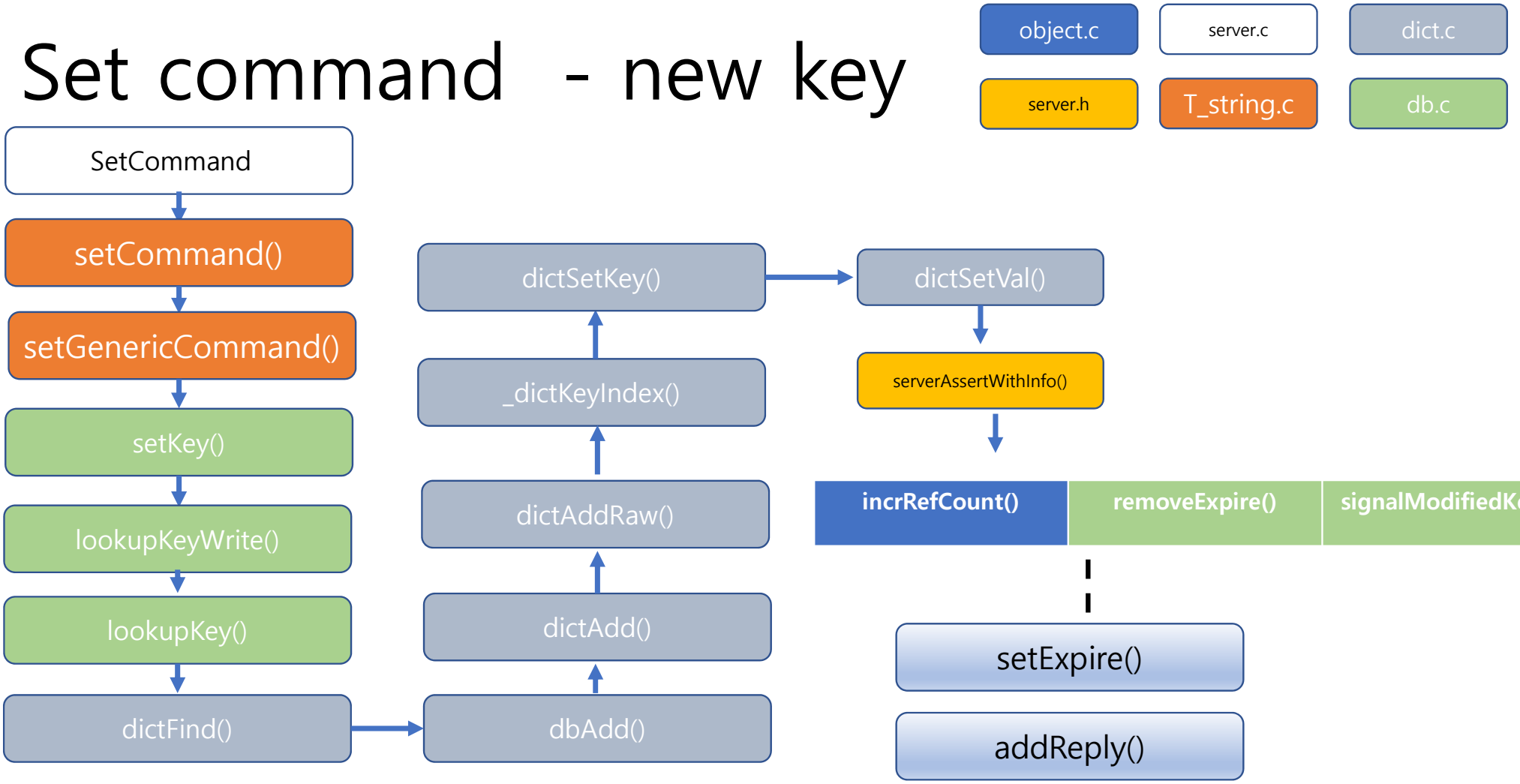

acceptTcpHandler()
anetTcpAccept()
anetGenericAccept()
acceptCommonHandler()
createClient()
aeCreateFileEvent()
-readyQueryFromClient
proc()
call()
lookupcommand()
processCommand()
ProcessInputbuffer()
aeApiAddEvent()

Get command


getCommand
getCommand()
addReply()
getGenericCommand()
dictGetVal()
lookupKeyReadOrReply()
lookupKey()
lookupKeyRead()
dictFind()
lookupKeyReadWithFlags()
lookupKey()


server.c
dict.c
T_string.c
db.c

Set command - new key



Set command – overwrite