# 라이브러리 설명

Shm 을 이용하여 전역메모리 사용

# 특이사항

* Ipc 중에서 가장 빠른 ipc
  + 구글에서 검색해보면, 각종 signal 이나 fd 통신을 이용한 ipc 보다 빠르다고 한다.
* 강제로 해제전까지 계속적으로 사용가능
  + 재부팅전까지 계속적으로 커널메모리에 상주
* Block 이 걸리지 않는다.
* 동기화이슈가 있다.
  + 각기 다른 application 에서 동시접근하기 때문에 동기화 이슈
  + 만약 동기화를 위해서는 mutex 를 사용한다.
    - Mutex handler 를 global 변수에 세팅후 동일한 핸들러로 mutex 구현가능

## 커널에 할당된 shm 확인하기

/data # ipcs -a

------ Shared Memory Segments --------

key shmid owner perms bytes nattch status

0x414e4547 0 root 644 65536 1

0x00001235 32769 root 666 14124 0

------ Semaphore Arrays --------

key semid owner perms nsems

0x414e4547 0 root 777 2

------ Message Queues --------

key msqid owner perms used-bytes messages

* APP\_SHM\_KEY\_PER\_APP 사용시 해당 id로 확인가능

## 커널의 shm 사이즈 확인 및 수정

# cat /proc/sys/kernel/shmmax

4278190079

# 사용중인 shm id

mds\_shm.h

#define APP\_SHM\_KEY\_GLOBAL 0x1234

#define APP\_SHM\_KEY\_PER\_APP 0x1235

#define PER\_APP\_MAX\_CNT 10

* APP\_SHM\_KEY\_GLOBAL : 프로그램에서 공용으로 사용할 메모리
* APP\_SHM\_KEY\_PER\_APP : 각 프로그램마다 할당된 메모리 (최대지원갯수는 PER\_APP\_MAX\_CNT에서 결정)

# API 설명

## Global 메모리 사용

전역 메모리 정의

typedef struct shm\_global\_data

{

int test\_int;

char msg\_buff[1024];

}\_\_attribute\_\_((packed))SHM\_GLOBAL\_DATA\_T;

* 전역메모리정의
* 해당 구조체는 변경하여사용가능.
  + 변경시 모든 어플을 재 컴파일 해서 사용해야 한다.

전역메모리 세팅

int app\_shm\_set\_global\_data(SHM\_GLOBAL\_DATA\_T\* data);

* 함수설명
  + 전역 메모리를 설정한다.
  + 메모리 설정후에는 UDP IPC 를 no-wait 로 전송한다.
    - UDP\_IPC\_PORT\_\_CHK\_GLOBAL\_MEM
  + 모든 application 에서 동일한 structure 로 데이터 세팅
* Return
  + MDS\_SHM\_TRUE : 설정
  + MDS\_SHM\_FALSE : 실패
* Argurment
  + data : global 메모리 구조체 포인터

전역메모리 얻기

int app\_shm\_get\_global\_data(SHM\_GLOBAL\_DATA\_T\* data);

* 함수설명
  + 전역 메모리를 얻어온다.
* Return
  + MDS\_SHM\_TRUE : 설정
  + MDS\_SHM\_FALSE : 실패
* Argurment
  + data : global 메모리 구조체 포인터

전역메모리 프린트하기

int app\_shm\_dbg\_print\_global\_data();

* 함수설명
  + 전역메모리를 프린트한다.
  + 전역메모리 구조체 변경시 해당 함수도 변경해야한다.
* Return
  + MDS\_SHM\_TRUE : 설정
  + MDS\_SHM\_FALSE : 실패

## 예제소스

SHM\_GLOBAL\_DATA\_T test\_global = {0,};

if (app\_shm\_get\_global\_data(&test\_global) == MDS\_SHM\_TRUE)

{

printf("get global mem success!\r\n");

app\_shm\_dbg\_print\_global\_data();

}

else

{

printf("get global mem fail! \r\n");

app\_shm\_dbg\_print\_global\_data();

}

test\_global.test\_int++;

if ( app\_shm\_set\_global\_data(&test\_global) == MDS\_SHM\_TRUE )

{

printf("set global mem success! \r\n");

app\_shm\_dbg\_print\_global\_data();

}

else

{

printf("set global mem fail! \r\n");

app\_shm\_dbg\_print\_global\_data();

}

## App 메모리사용

* 각 application 마다 각각의 메모리 공간을 할당하여 사용가능 하다.
* 이때 전역 메모리처럼 각각의 어플리케이션 메모리 영역을 모든 어플리케이션에서 접근 가능
* 각 어플리케이션의 메모리 영역에 대한 분류는 스트링으로 한다.
  + 종료되었다가 다시 실행되더라도… 해당 appname으로 접근하면 다시 데이터 확인 가능
* 최대 10개 app 메모리 공간 지원
  + PER\_APP\_MAX\_CNT

App 메모리 정의

typedef struct shm\_per\_app\_data

{

char app\_id[128]; // don`t edit!!!

int test\_int;

unsigned char msg\_buff[1024];

}\_\_attribute\_\_((packed))SHM\_PER\_APP\_DATA\_T;

* 각각의 어플리케이션 메모리 영역
* 특히 app\_id 는 수정하지 말아야한다.
  + 모든 어플의 메모리를 얻어올 경우 어떤 어플용 메모리를 얻어 왔는지 확인하기 위한 용도.

App 메모리 설정

int app\_shm\_set\_per\_app\_data(char\* app\_name, SHM\_PER\_APP\_DATA\_T\* data);

* 함수설명
  + App 메모리를 설정한다.
* Return
  + MDS\_SHM\_TRUE : 설정
  + MDS\_SHM\_FALSE : 실패
* Argument
  + app\_name : 각 메모리 설정을 위한 id : string
  + data : app 메모리 구조체 포인터

App 메모리 획득

int app\_shm\_get\_per\_app\_data(char\* app\_name, SHM\_PER\_APP\_DATA\_T\* data);

* 함수설명
  + App 메모리를 얻어온다.
* Return
  + MDS\_SHM\_TRUE : 설정
  + MDS\_SHM\_FALSE : 실패
* Argument
  + app\_name : 각 메모리 설정을 위한 id : string
  + data : app 메모리 구조체 포인터

모든 app 메모리 획득

int app\_shm\_get\_per\_app\_data\_all(SHM\_PER\_APP\_DATA\_T (\*data)[PER\_APP\_MAX\_CNT]);

* 함수설명
  + 모든 App 메모리를 얻어온다.
* Return
  + 총 app 메모리 설정된 개수
* Argument
  + app\_name : 각 메모리 설정을 위한 id : string
  + data : app 메모리 구조체 포인터

## 예제소스

{

int read\_cnt = 0;

int i = 0;

SHM\_PER\_APP\_DATA\_T per\_app\_data\_array[PER\_APP\_MAX\_CNT];

SHM\_PER\_APP\_DATA\_T per\_app\_temp;

if (argc < 2)

{

printf("need to arg!!\r\n");

exit(0);

}

printf("program start...\r\n");

if ( app\_shm\_get\_per\_app\_data(argv[1], &per\_app\_temp) == MDS\_SHM\_TRUE )

{

printf("get per app mem success! \r\n");

printf("per\_app\_temp->test\_int is [%d] \r\n", per\_app\_temp.test\_int);

}

else

{

printf("get per app mem fail! \r\n");

}

per\_app\_temp.test\_int++;

if ( app\_shm\_set\_per\_app\_data(argv[1], &per\_app\_temp) == MDS\_SHM\_TRUE )

{

printf("set per app mem success!\r\n");

}

else

{

printf("set per app mem fail ! \r\n");

}

memset(&per\_app\_data\_array, 0x00, sizeof(SHM\_PER\_APP\_DATA\_T)\*PER\_APP\_MAX\_CNT);

read\_cnt = app\_shm\_get\_per\_app\_data\_all(&per\_app\_data\_array);

printf(" ---------------------------- \r\n");

printf("app data all [%d] \r\n", read\_cnt);

for ( i = 0 ; i < read\_cnt ; i ++)

{

printf("per\_app\_data[%d].app\_id ==> [%s]\r\n", i, per\_app\_data\_array[i].app\_id);

printf("per\_app\_data[%d].test\_int ==> [%d]\r\n", i, per\_app\_data\_array[i].test\_int);

}

printf(" ---------------------------- \r\n");

}