在设备中缓存视频

录制视频流程

1. 在main函数中调用 `sample_dmc_init`

libdmc_pes.c | main.c 2 | libpes.c | 录制一分钟.md

src > main.c > main(int, char * [])

```c
int main(int argc, char *argv[])
    src.obj_id = FH_OBJ_ISP;
    src.dev_id = 0;
    src.chn_id = 0;
    dst.obj_id = FH_OBJ_VPU_VI;
    dst.dev_id = 0;
    dst.chn_id = 0;
    ret = FH_SYS_Bind(src, dst);
    CHECK_RET(ret != 0, ret);

    ret = FH_VENC_StartRecvPic(0);
    CHECK_RET(ret != 0, ret);

    src.obj_id = FH_OBJ_VPU_VO;
    src.dev_id = GROUP_ID;
    src.chn_id = 0;

    dst.obj_id = FH_OBJ_ENC;
    dst.dev_id = 0;
    dst.chn_id = 0;

    ret = FH_SYS_Bind(src, dst);
    CHECK_RET(ret != 0, ret);

    sample_dmc_init(dst_ip, port, 1);

    pthread_attr_t attr;
    pthread_t thread_stream;
```

2. 在 `sample_dmc_init` 函数中调用 `dmc_pes_subscribe`

libdmc_pes.c | main.c 2 | libpes.c | 录制一分钟.md

src > main.c > sample_dmc_init(FH_CHAR *, FH_UINT32, FH_SINT32)

```c

int sample_dmc_init(FH_CHAR *dst_ip, FH_UINT32 port ,FH_SINT32 max_channel_no)
{
    dmc_init();

    if (dst_ip != NULL && *dst_ip != 0)
    {
        dmc_pes_subscribe(max_channel_no, dst_ip, port);
    }

    return 0;
}
```

3. 在 `dmc_pes_subscribe` 函数中调用 `libpes_init`

libdmc_pes.c × | libpes.c 6 | 录制一分钟.md

src > libdmc_pes.c > dmc_pes_subscribe(int, char *, int)

```c
static int _pes_input_fn(int media_chn, int media_type, int media_subtype, \
    return 0;
}

int dmc_pes_subscribe(int max_channel, char* ip, int port)
{
    int i;
    int ret;

    if (ip == NULL)
    {
        printf("Error: NULL ip address, please run \"vlcview -h\"\n");
        return -1;
    }

    if (max_channel > MAX_PES_CHANNEL_COUNT)
    {
        printf("Error: channel num is larger than %d\n", MAX_PES_CHANNEL_COUNT);
        return -1;
    }

    g_frame_length = 0;
    g_nalu_count   = 0;

    ret = libpes_init();
    if (ret != 0)
    {
        return -1;
    }

    strncpy(g_print_info.tar_ip, ip, sizeof(g_print_info.tar_ip));

    for(i = 0; i < max_channel; i++)
    {
        libpes_send_to_vlc(i, ip, port + i);
        g_print_info.port[i] = port + i;
        g_print_info.printed[i] = 0;
    }

    dmc_subscribe("PES", DMC_MEDIA_TYPE_H264 | DMC_MEDIA_TYPE_H265, _pes_input_fn);
    return 0;
}
```

4. 在 `dmc_pes_subscribe` 函数中调用创建线程

libdmc_pes.c | main.c 2 | libdmc.c | libpes.c 6 | 录制一分钟.md

src > libpes.c > _VLC_UNLOCK()

```c
787 int libpes_init(void)
789     pthread_attr_t attr;
790     pthread_t thread;
791     int fd;
792     int i;
793
794     _VLC_LOCK();
795     if (!g_this_module_inited)
796     {
797         fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
798         if (fd < 0)
799         {
800             printf("Error: libpes failed with socket!\n");
801             _VLC_UNLOCK();
802             return -1;
803         }
804
805         g_stop_request = 0;
806
807         pthread_attr_init(&attr);
808         pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
809         pthread_attr_setstacksize(&attr, 3 * 1024);
810
811 #ifdef __RTTHREAD_OS__
812         struct sched_param param;
813         param.sched_priority = FH_PES_THREAD_PRORITY;
814         pthread_attr_setschedparam(&attr, &param);
815 #endif
816
817         if (pthread_create(&thread, &attr, network_send_proc, (void*)fd) != 0)
818         {
819             printf("Error: Create libpes thread failed!\n");
820             _VLC_UNLOCK();
821             return -1;
```

设置为缓存一分钟视频

`在libpes.c` 中找到 `int libpes_init(void)`， 位于第787行

在其中添加定时器函数

```c
// 注册信号处理函数
signal(SIGALRM, alarm_handler);

// 设置定时器为1分钟
alarm(60);
```

`alarm(60)` 代表在六十秒后，执行函数 `void alarm_handler(int signum)`：

```c
void alarm_handler(int signum)
{
    if (signum == SIGALRM)
    {
        fclose(ps_file);
        close(fd);
        g_stop_request = 0;
        printf("\n****************************************\n \
        One minute has passed. Exiting...\n");
        exit(EXIT_SUCCESS);
    }
}
```

因为设备在推流视频时，同时将视频保存在了本地，这里`fclose`后，会把视频保存在`/home/h264.ps`，随后将其复制出即可

P.S. 我把`static FILE *ps_file = NULL;`设置成了全局变量，这样可以在不同的函数里面调用