PA 3.3 访存越界

在完成 PA 3.3的时候，出现了访存越界的现象，整个调试过程中difftest都是开着的：

```
address (0x00000015) is out of bound at pc = 0x8301b7f0
$0 = 0x00000000
ra = 0x8301be78
sp = 0x80379e90
gp = 0x00000000
tp = 0x00000000
t0 = 0x00000000
t1 = 0x8301b4cc
t2 = 0x00000000
s0 = 0x83029b0c
s1 = 0x00000015
a0 = 0x00000015
a1 = 0x0000000e
a2 = 0x00000000
a3 = 0x80379edc
a4 = 0x83029b0c
a5 = 0x00000015
a6 = 0x83029700
a7 = 0x00000004
s2 = 0x80379edc
s3 = 0x00000016
s4 = 0x00000000
s5 = 0x00000000
s6 = 0x00000000
s7 = 0x00000000
s8 = 0x00000000
s9 = 0x00000000
s10 = 0x00000000
s11 = 0x00000000
t3 = 0x00000000
t4 = 0x83021f80
t5 = 0x00000000
t6 = 0x00000000
```

通过print调试法最终确定在 `NDL_OpenCanvas` 函数中:

```c
void NDL_OpenCanvas(int *w, int *h) {
    // 打开 /proc/dispinfo 文件
    int dispinfo_fd = open(file: "/proc/dispinfo", oflag: O_RDONLY);
    printf(format: "After open\n");


    // 读取文件内容
    size_t bytes_read = read(fd: dispinfo_fd, buf: dispinfo_buf, nbytes: sizeof(dispinfo_buf));

    printf(format: "After read\n");
```

```
After open
Just in _read
After _syscall_ read
address (0x00000015) is out of bound at pc = 0x8301b7f0
```

然后继续跟到 `read` 系统调用中，发现read应该是正常返回的：

```
int _read(int fd, void *buf, size_t count) {
  printf(format: "Just in _read\n");
  int ret = _syscall_(type: SYS_read, a0: fd, a1: (intptr_t)buf, a2: count);
  printf(format: "After _syscall_ read\n");
  return ret;
}
```

```
After open
Just in _read
After _syscall_ read
address (0x00000015) is out of bound at pc = 0x8301b7f0
```

这就很奇怪了，如果C库是正确的，那就应该回到`NDL_OpenCanvas`函数中，但是并没有

接下来应该怎么调试？