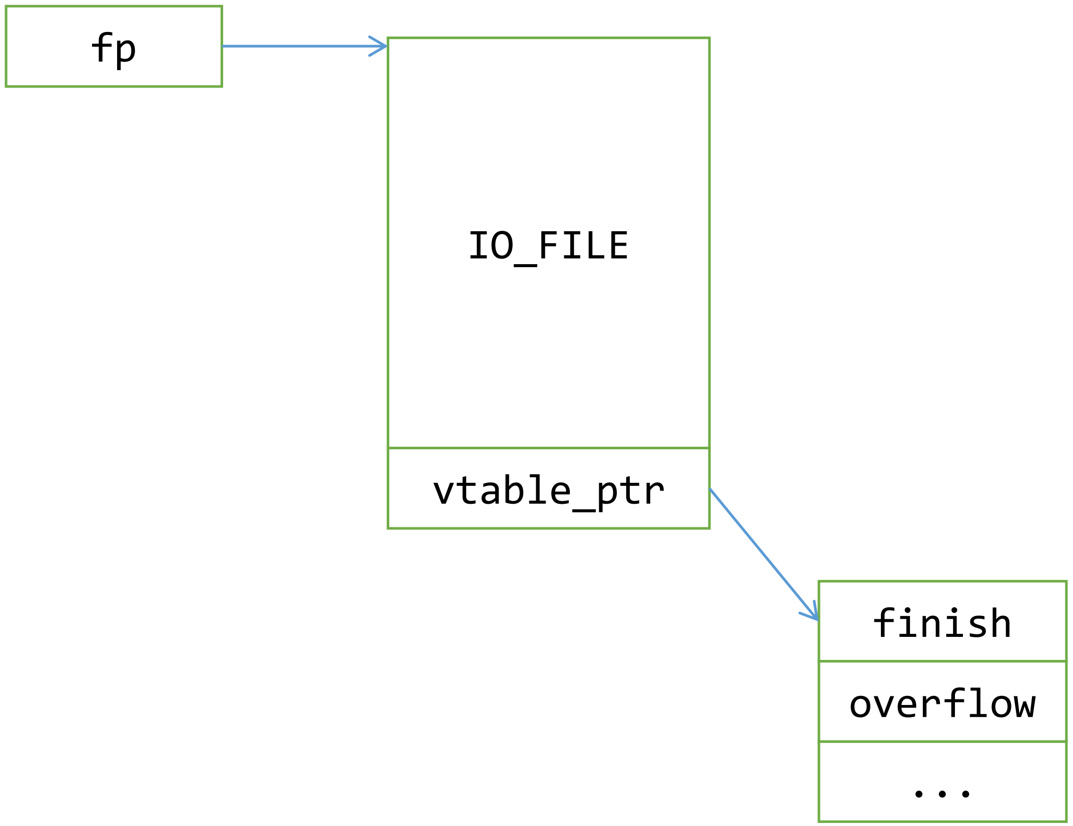


fp

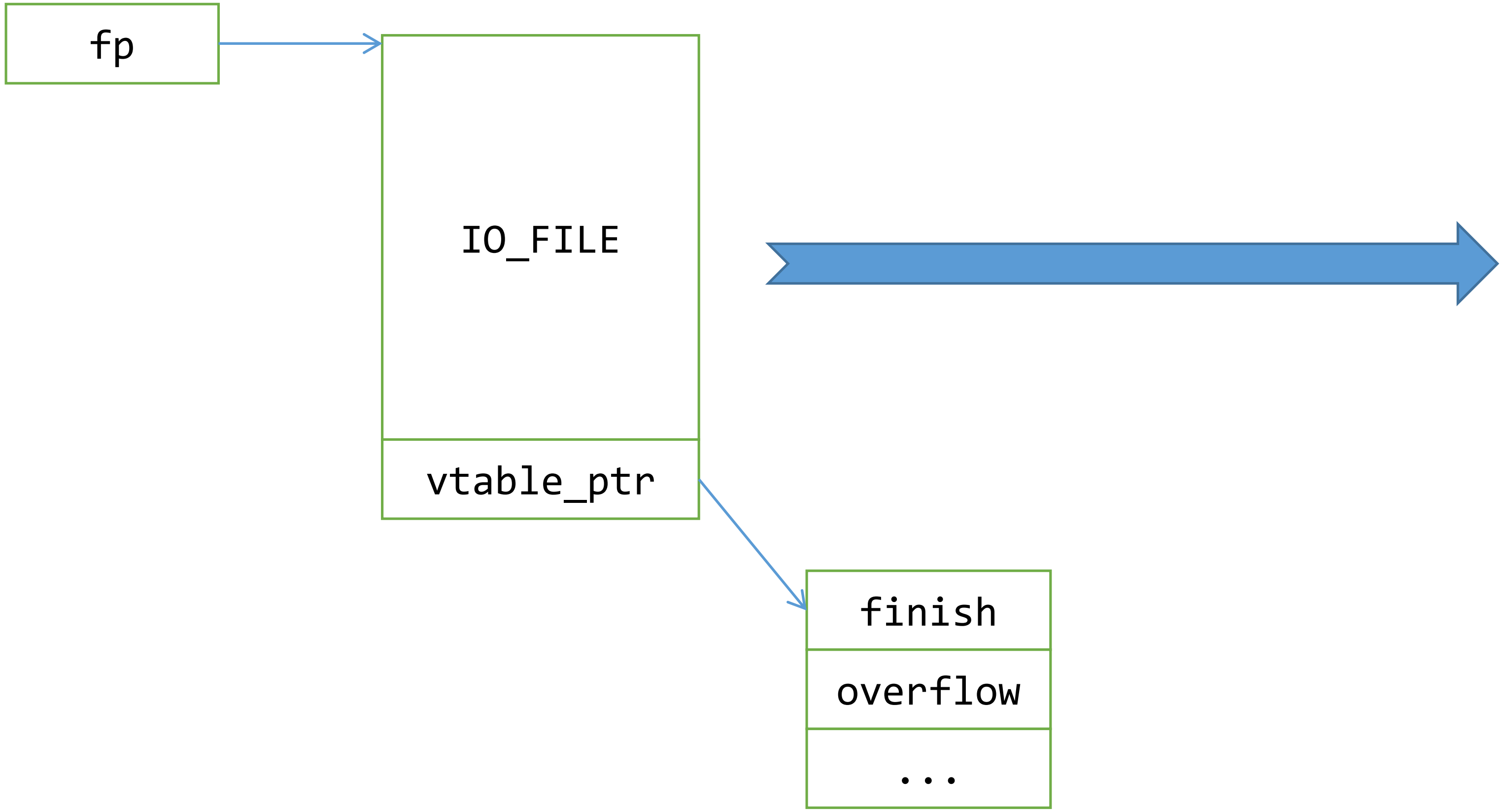
IO_FILE

vtable_ptr

finish

overflow

...



fp

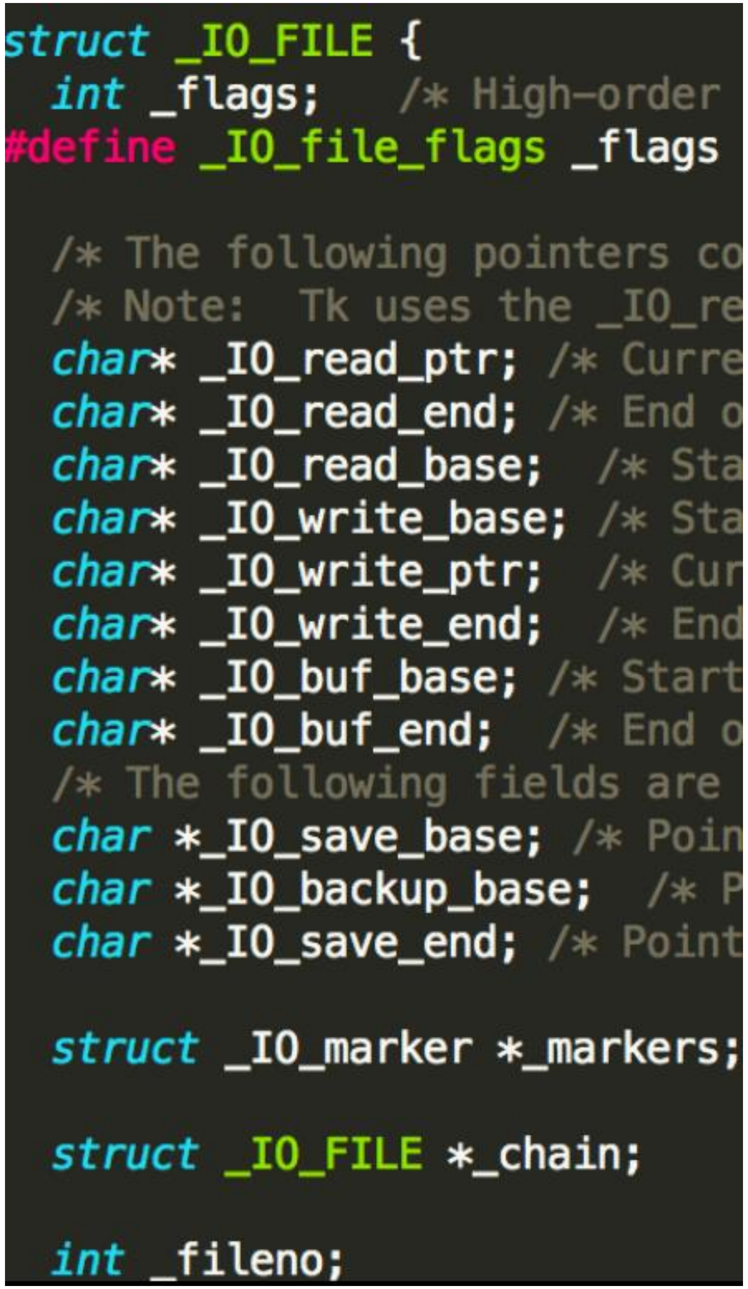
IO_FILE

vtable_ptr

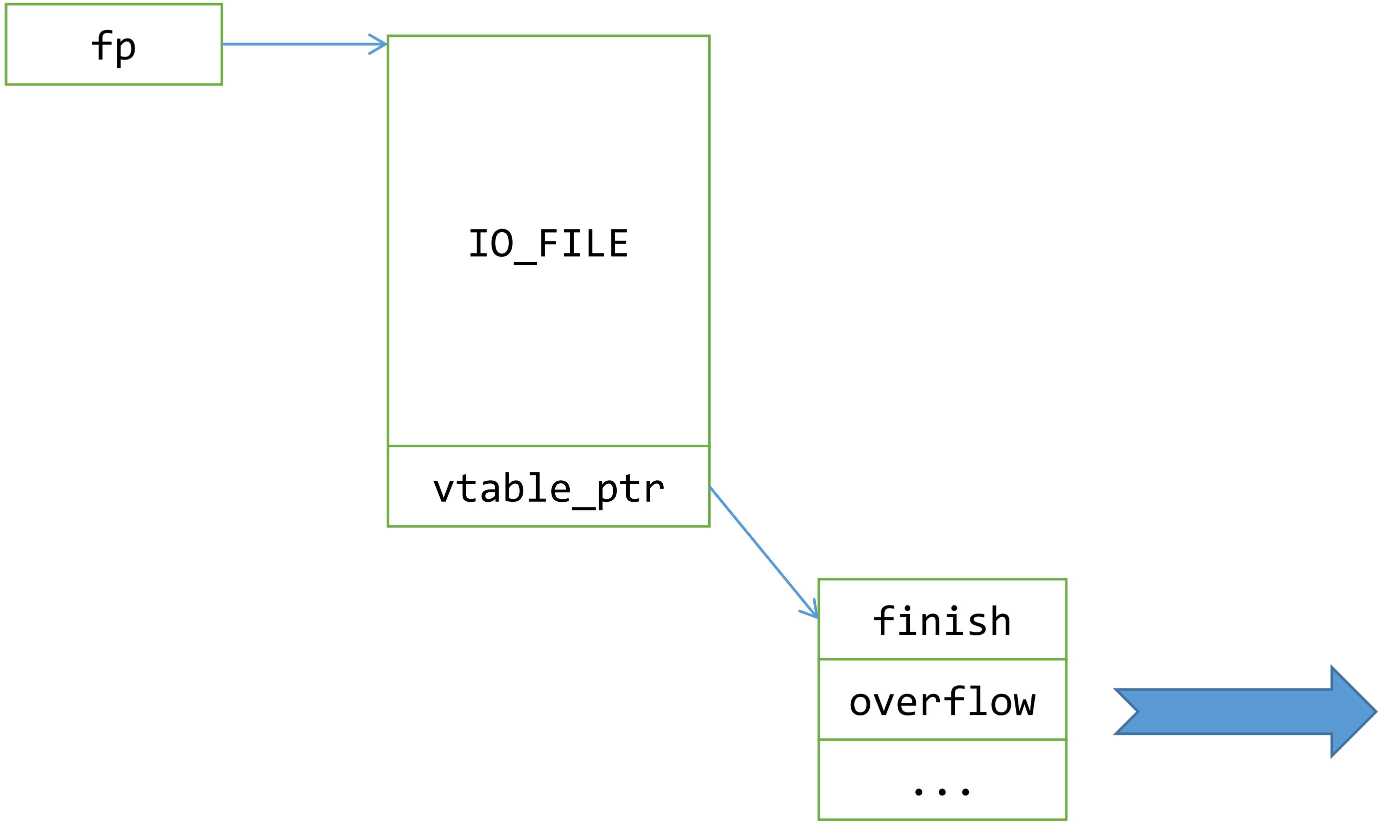
finish

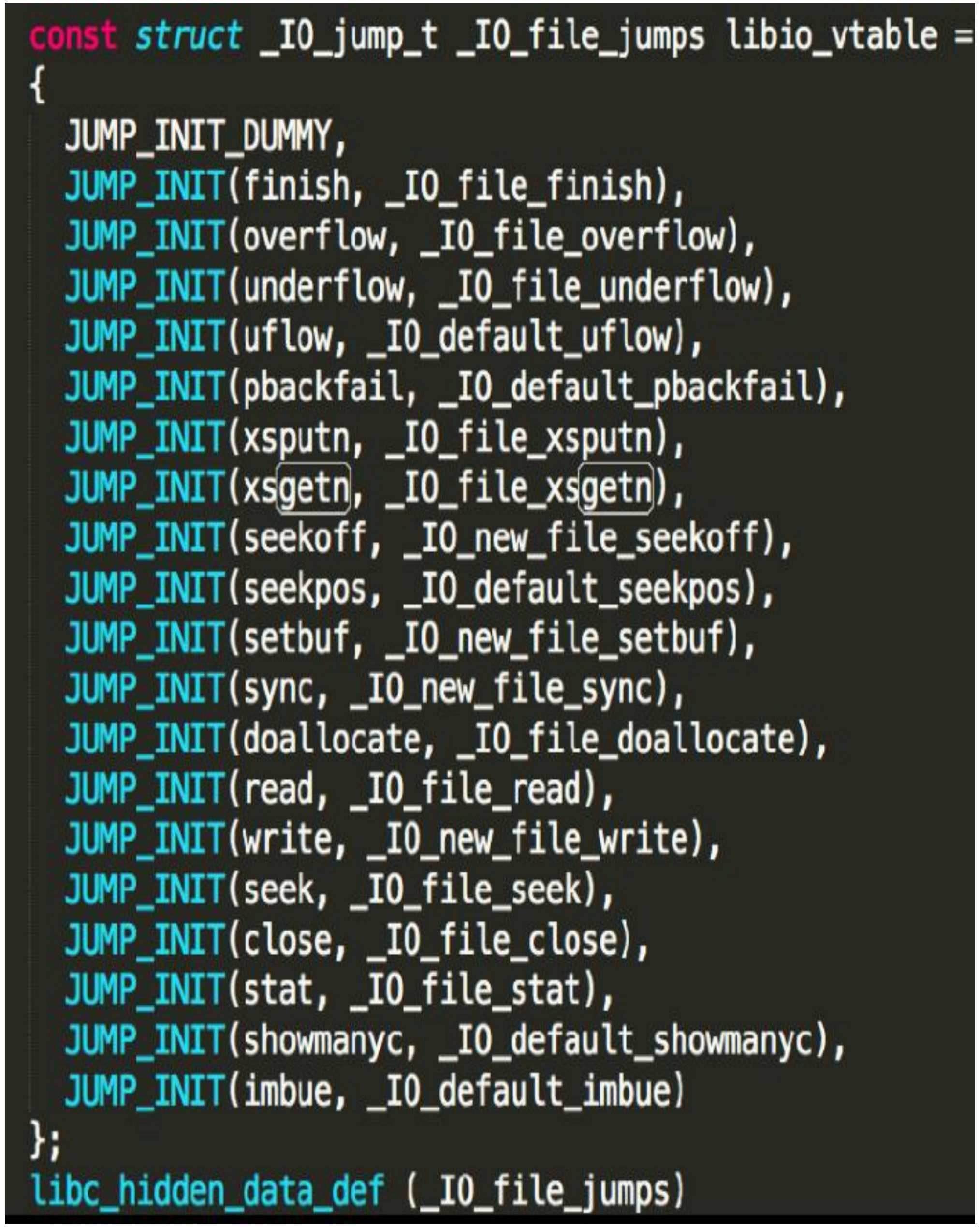
overflow

...

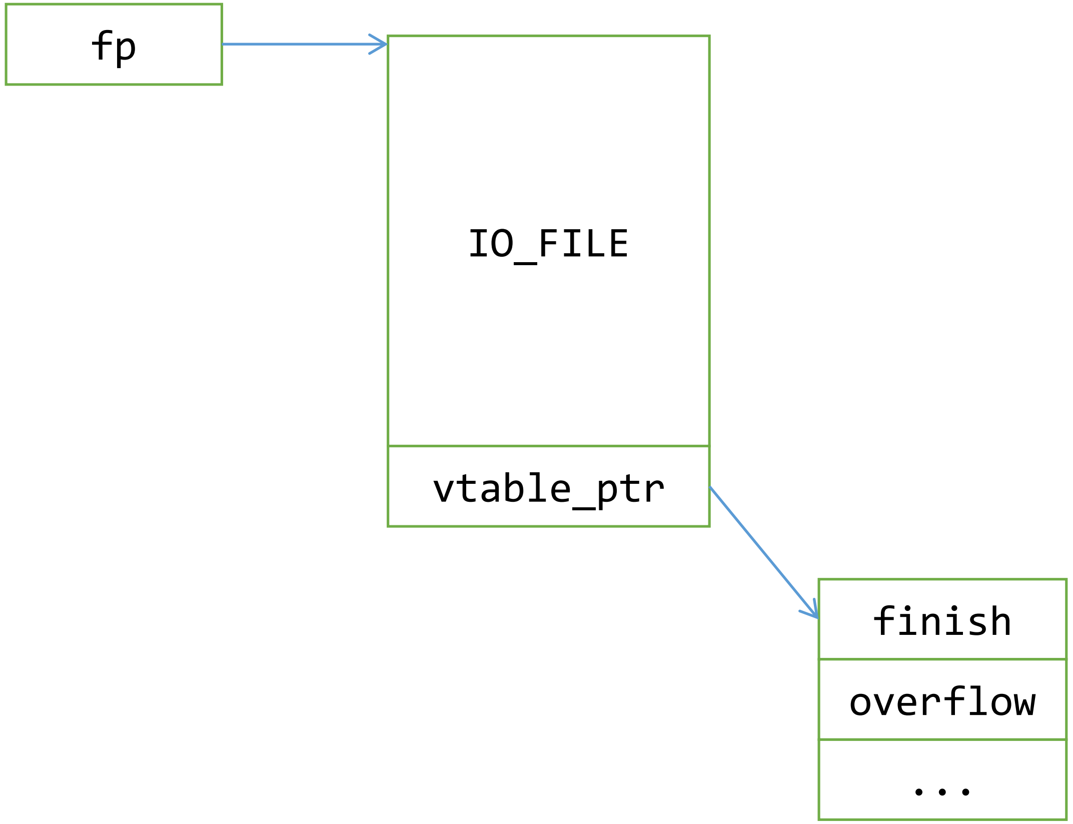


```
struct _IO_FILE {
  int _flags;    /* High-order
#define _IO_file_flags _flags

  /* The following pointers co
  /* Note:  Tk uses the _IO_re
  char* _IO_read_ptr; /* Curre
  char* _IO_read_end; /* End o
  char* _IO_read_base;  /* Sta
  char* _IO_write_base; /* Sta
  char* _IO_write_ptr;  /* Cur
  char* _IO_write_end;  /* End
  char* _IO_buf_base; /* Start
  char* _IO_buf_end;  /* End o
  /* The following fields are
  char *_IO_save_base; /* Poin
  char *_IO_backup_base;  /* P
  char *_IO_save_end; /* Point

  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;

  int _fileno;
```





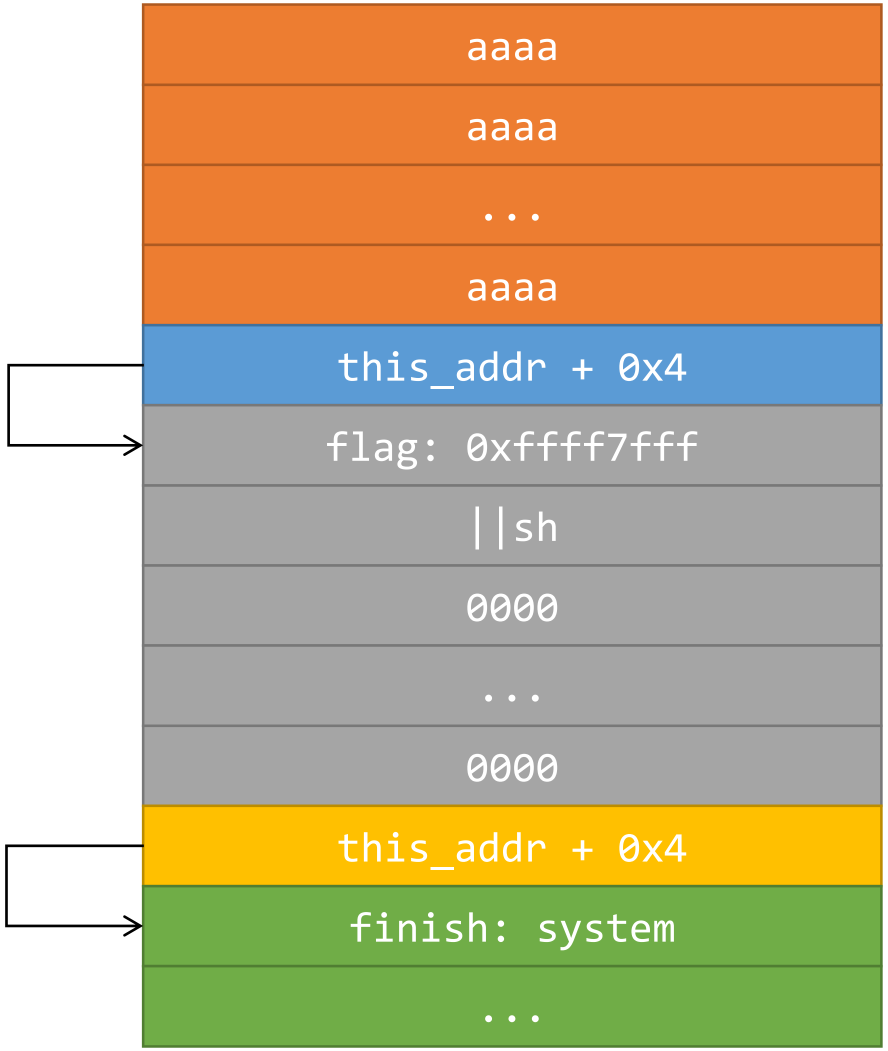
```
const struct _IO_jump_t _IO_file_jumps libio_vtable =
{
  JUMP_INIT_DUMMY,
  JUMP_INIT(finish, _IO_file_finish),
  JUMP_INIT(overflow, _IO_file_overflow),
  JUMP_INIT(underflow, _IO_file_underflow),
  JUMP_INIT(uflow, _IO_default_uflow),
  JUMP_INIT(pbackfail, _IO_default_pbackfail),
  JUMP_INIT(xsputn, _IO_file_xsputn),
  JUMP_INIT(xsgetn, _IO_file_xsgetn),
  JUMP_INIT(seekoff, _IO_new_file_seekoff),
  JUMP_INIT(seekpos, _IO_default_seekpos),
  JUMP_INIT(setbuf, _IO_new_file_setbuf),
  JUMP_INIT(sync, _IO_new_file_sync),
  JUMP_INIT(doallocate, _IO_file_doallocate),
  JUMP_INIT(read, _IO_file_read),
  JUMP_INIT(write, _IO_new_file_write),
  JUMP_INIT(seek, _IO_file_seek),
  JUMP_INIT(close, _IO_file_close),
  JUMP_INIT(stat, _IO_file_stat),
  JUMP_INIT(showmanyc, _IO_default_showmanyc),
  JUMP_INIT(imbue, _IO_default_imbue)
};
libc_hidden_data_def (_IO_file_jumps)
```

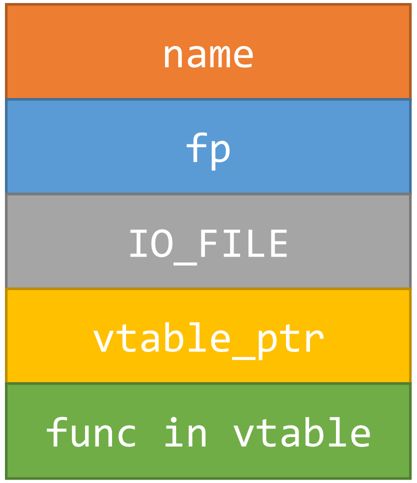


fclose() 步骤

利用IO_FILE.flag和其他一些东西
如果符合条件，就执行vtable.finish(fp)

这题我没有深究
flag不变就可以执行vtable.finish(fp)
具体作用以后再研究





最终执行finish(fp)

等于 system(“0xffff7fff||sh”)