

238P Operating Systems, Fall 2018

Final Review **Call Chain of a System Call**

7 December 2018

Aftab Hussain

University of California, Irvine

a visual summary of what
we've done thus far

xv6 is a program

*a legend of the diagrams to
follow*

tool/method

an xv6
functionality

T

topic

FRONT END

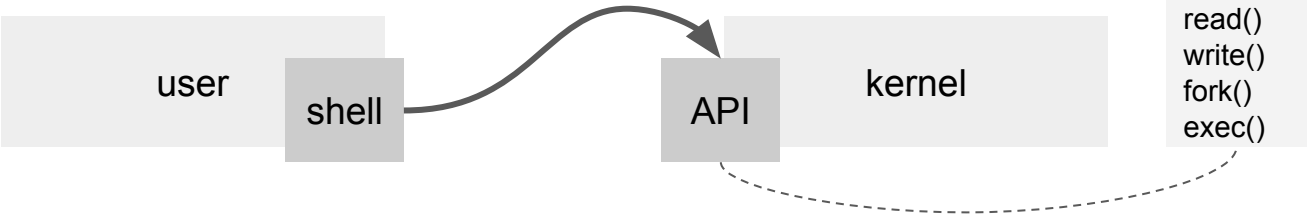
user

kernel

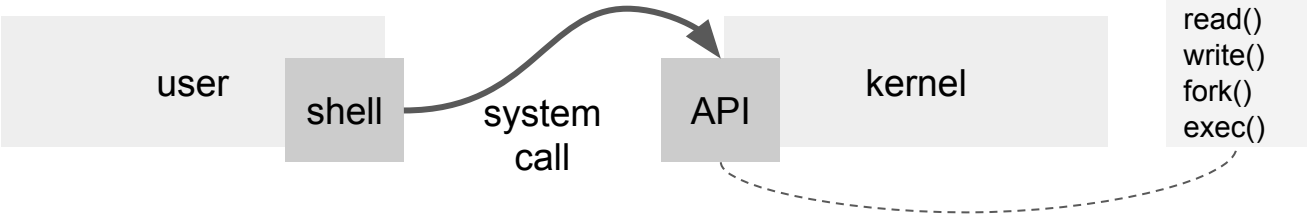
FRONT END



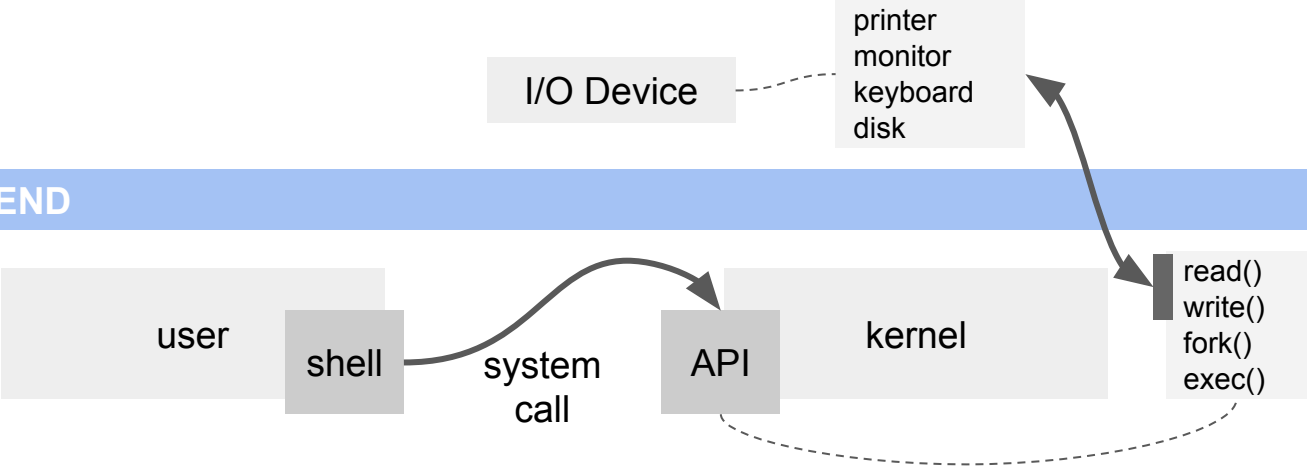
FRONT END



FRONT END



FRONT END



I/O Device

printer
monitor
keyboard
disk

user

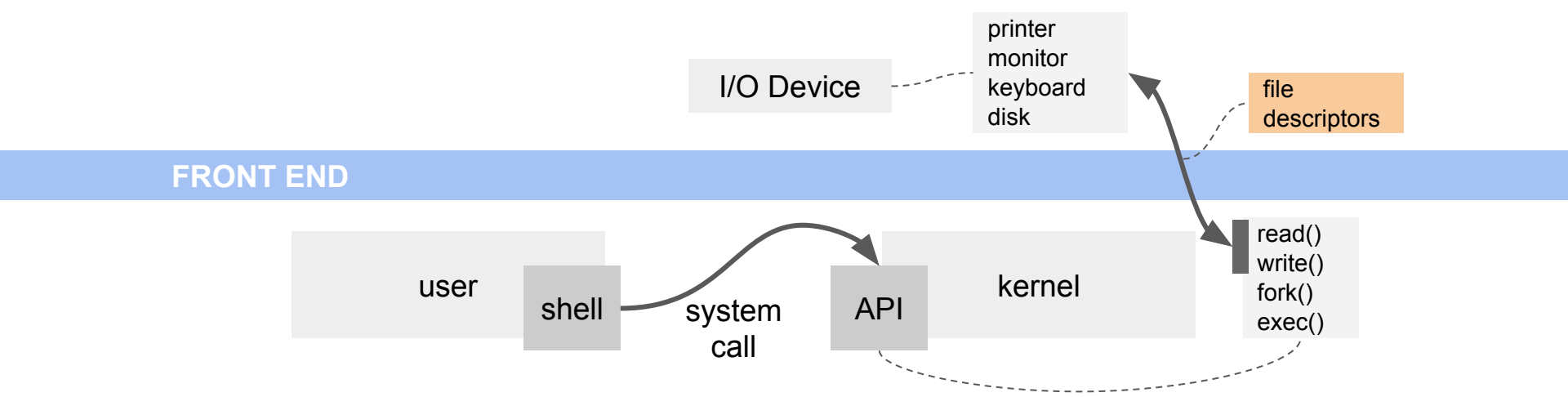
shell

system
call

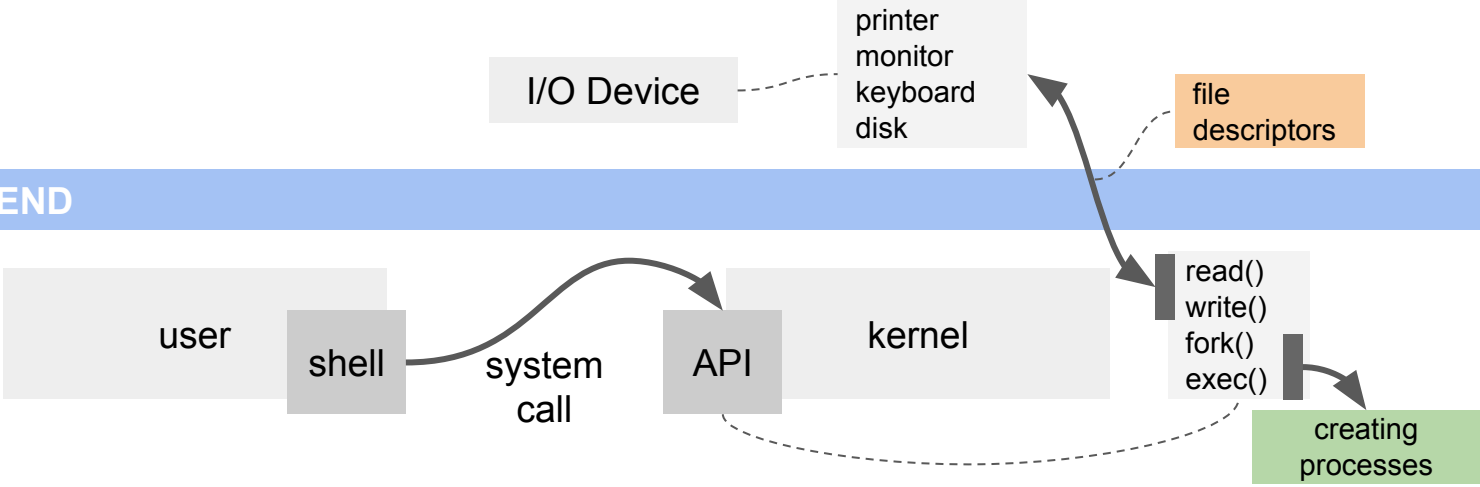
API

kernel

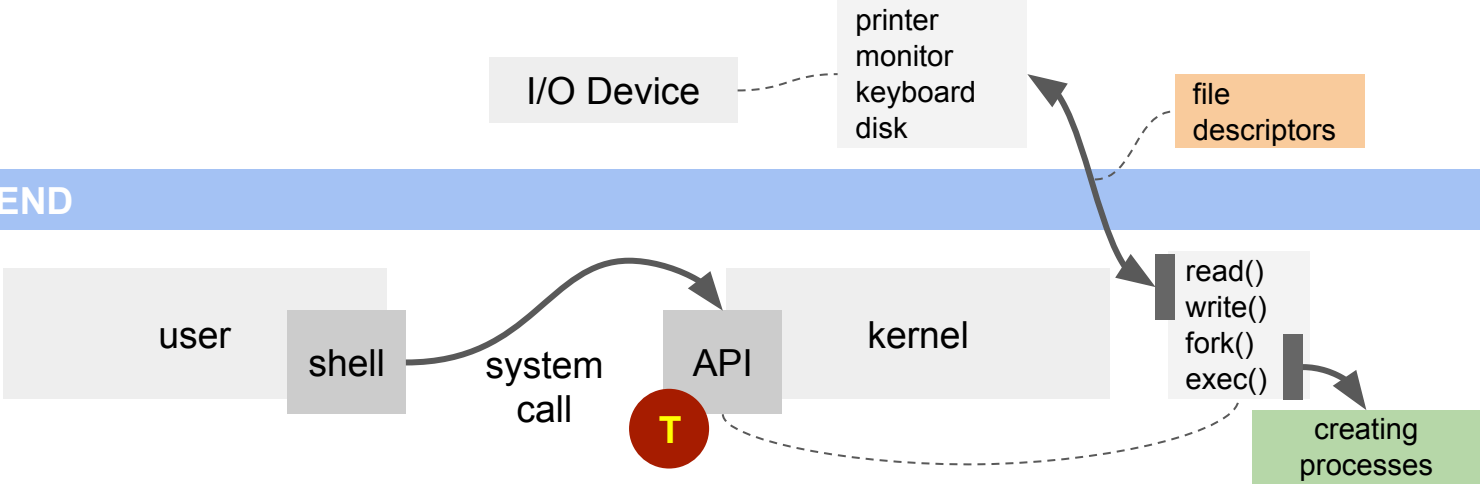
read()
write()
fork()
exec()



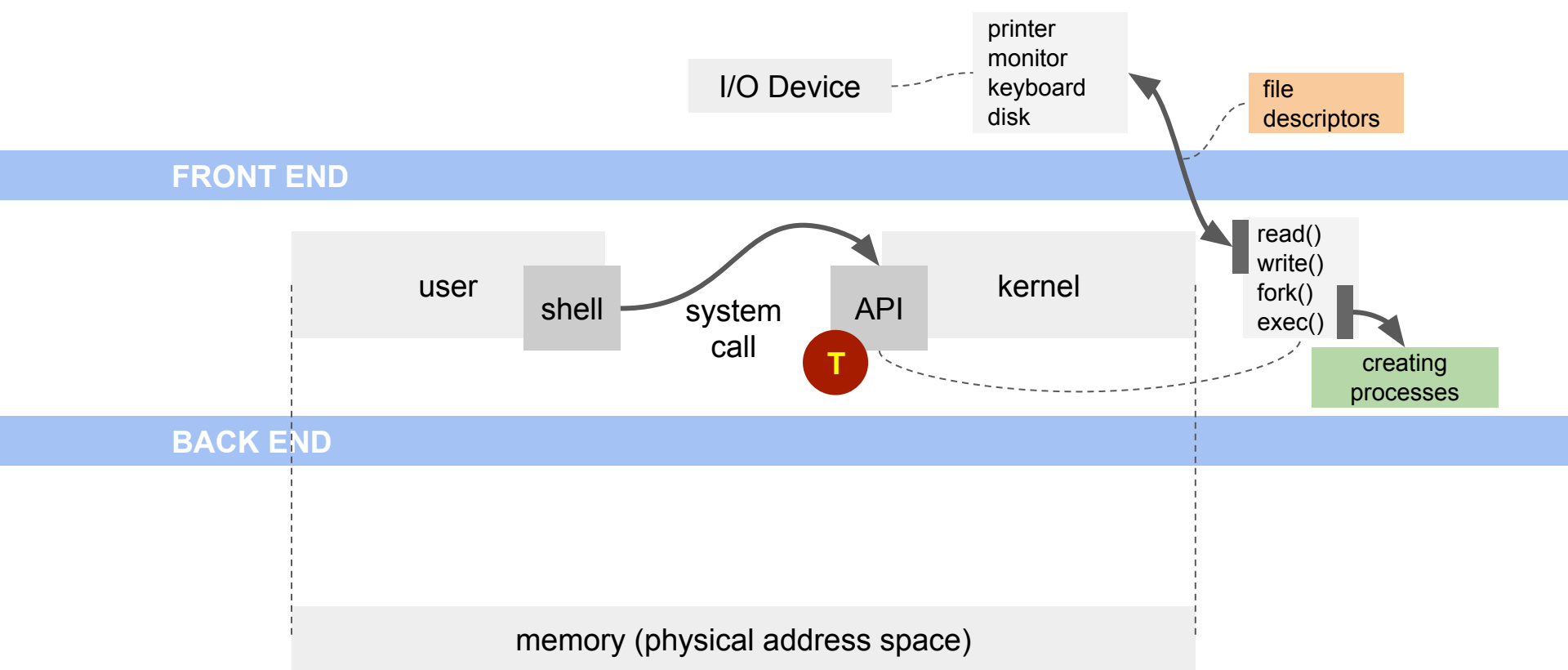
FRONT END

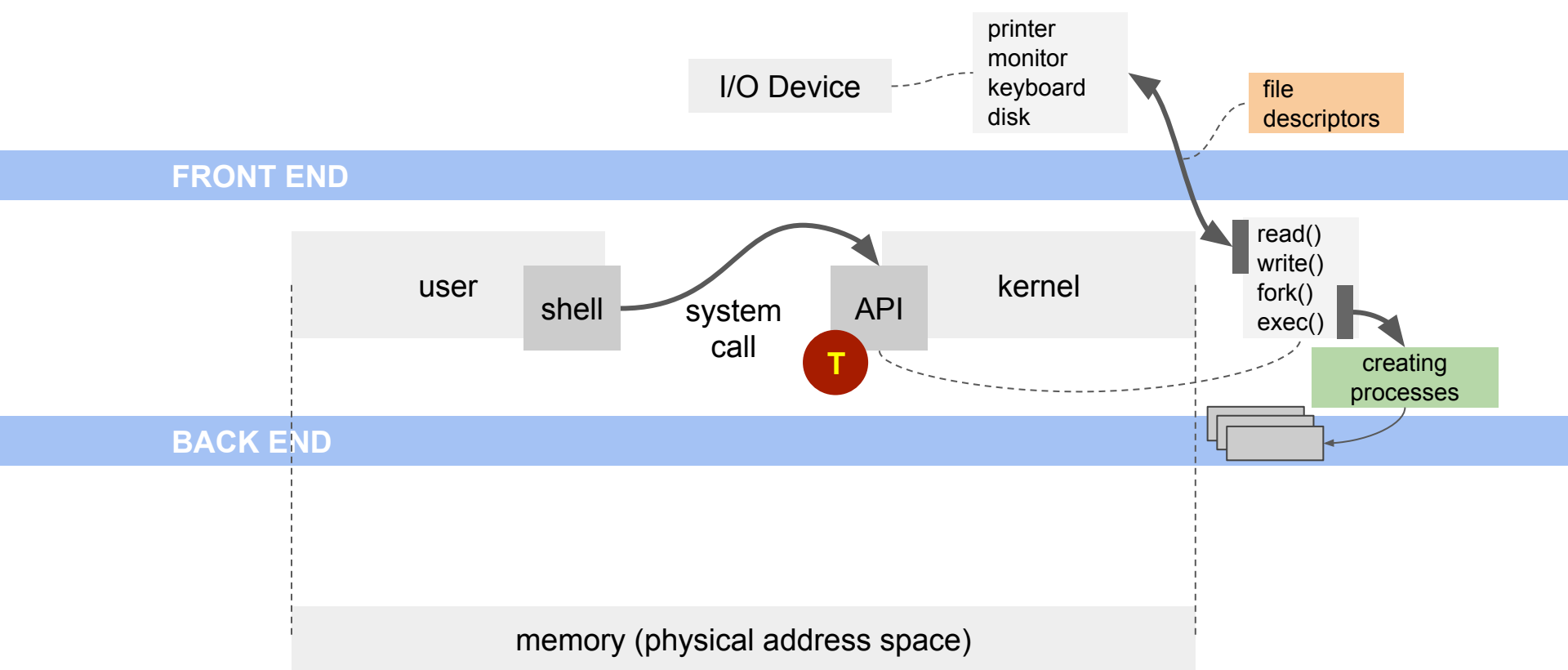


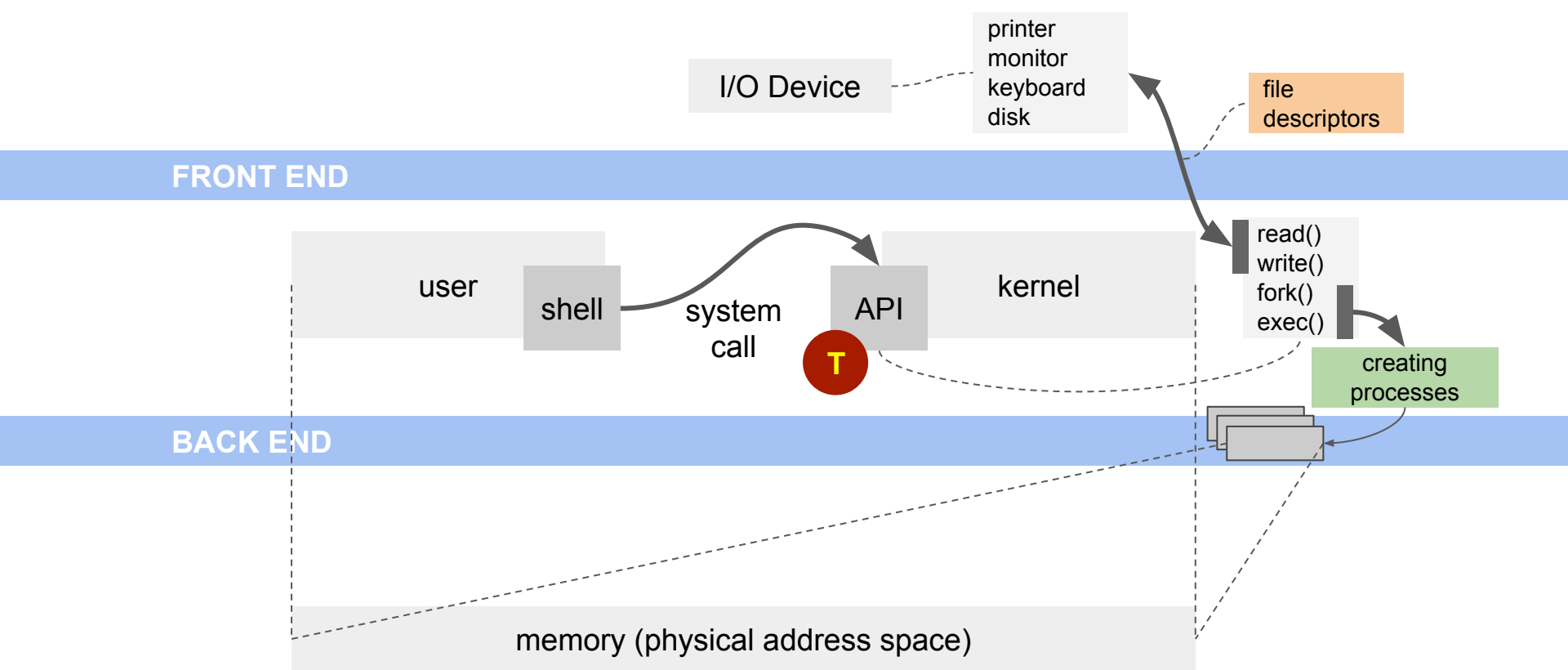
FRONT END

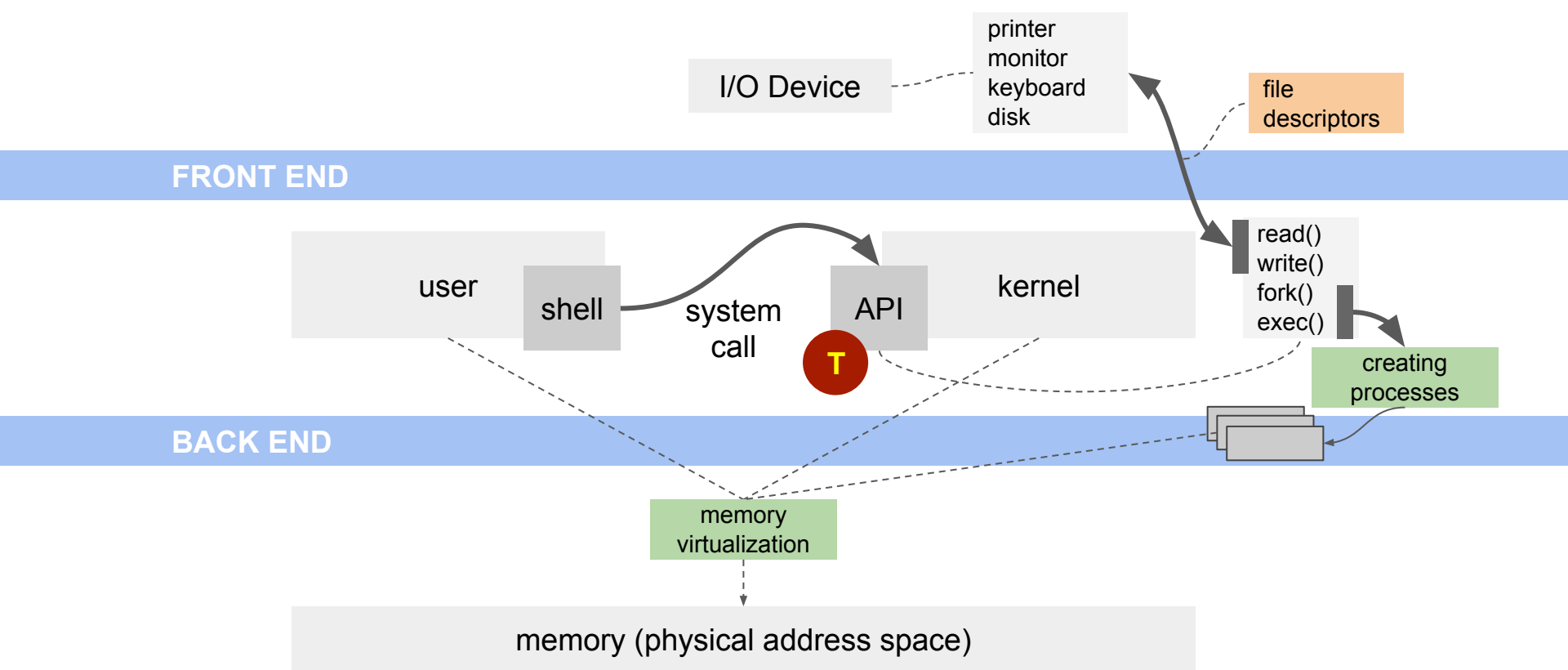


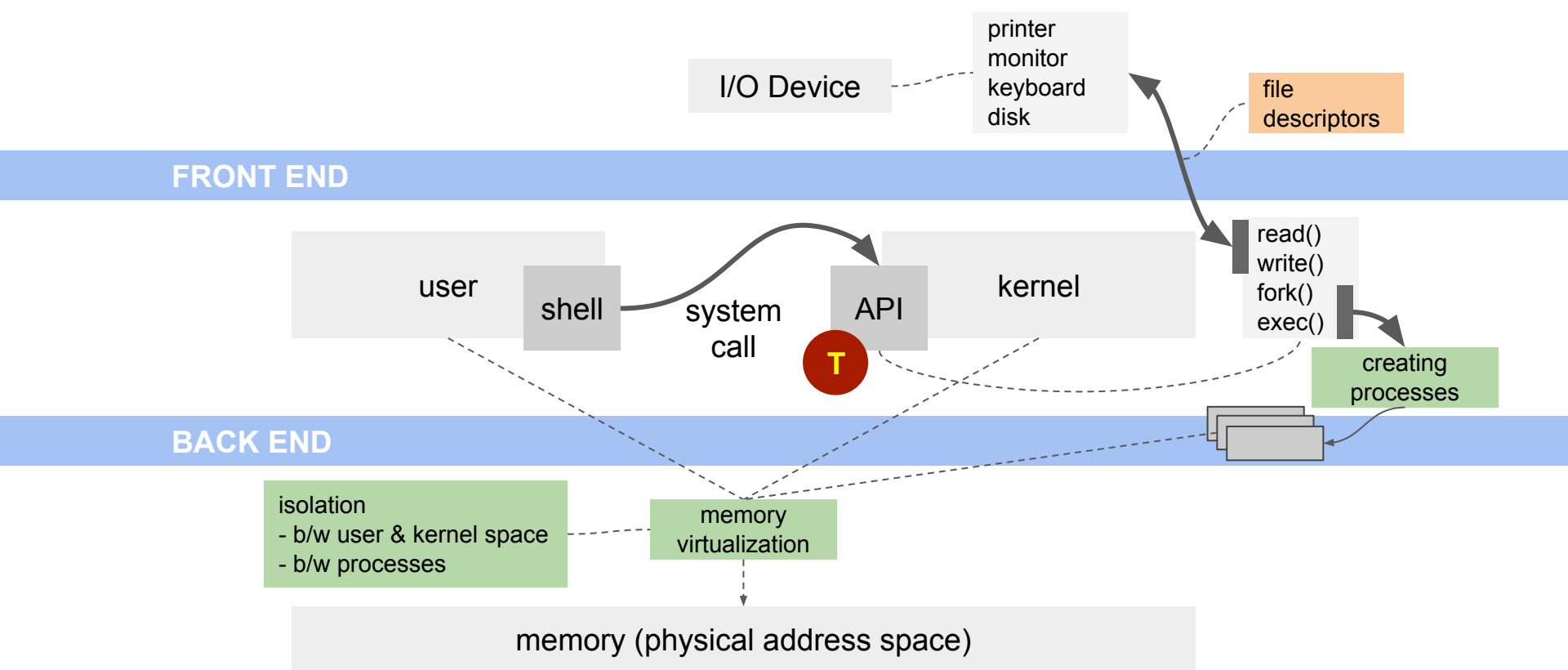
Any running program -- i.e., *a process*
-- must be in memory, so the CPU can
run it.

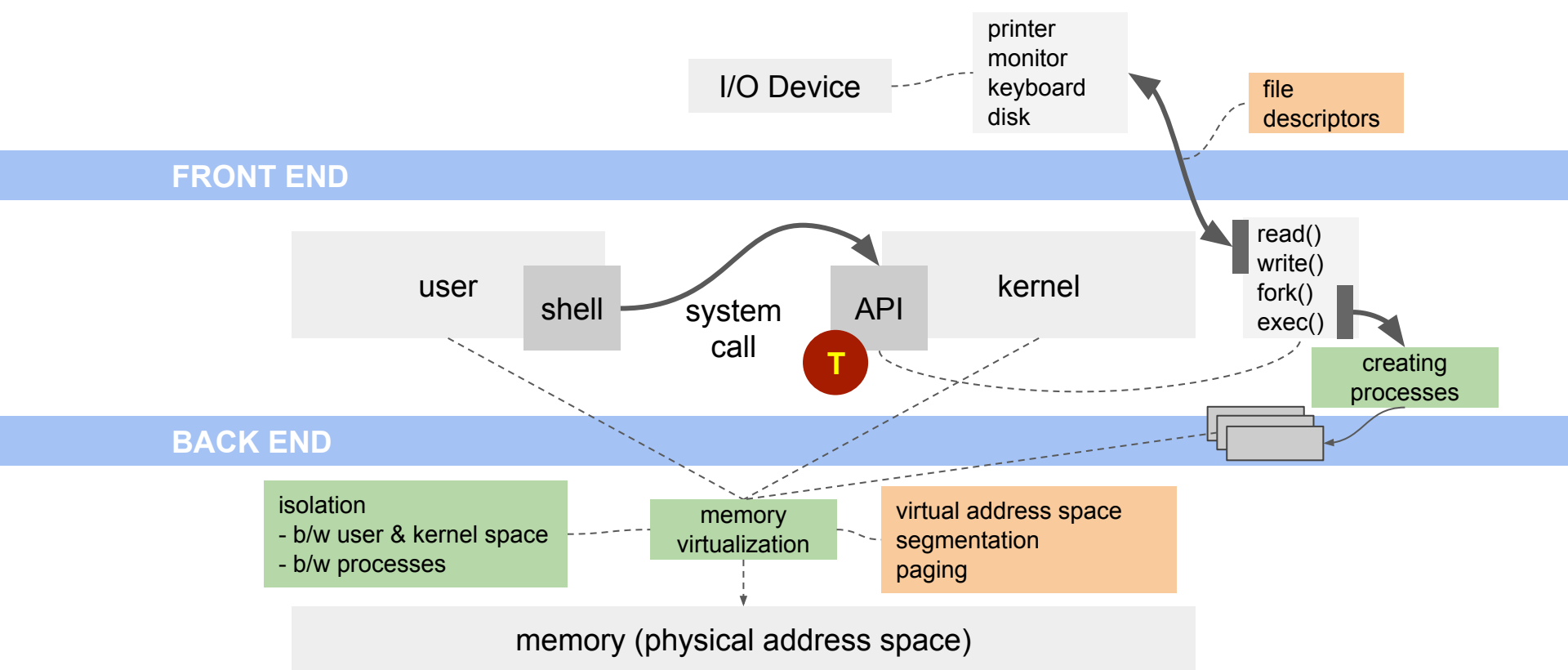


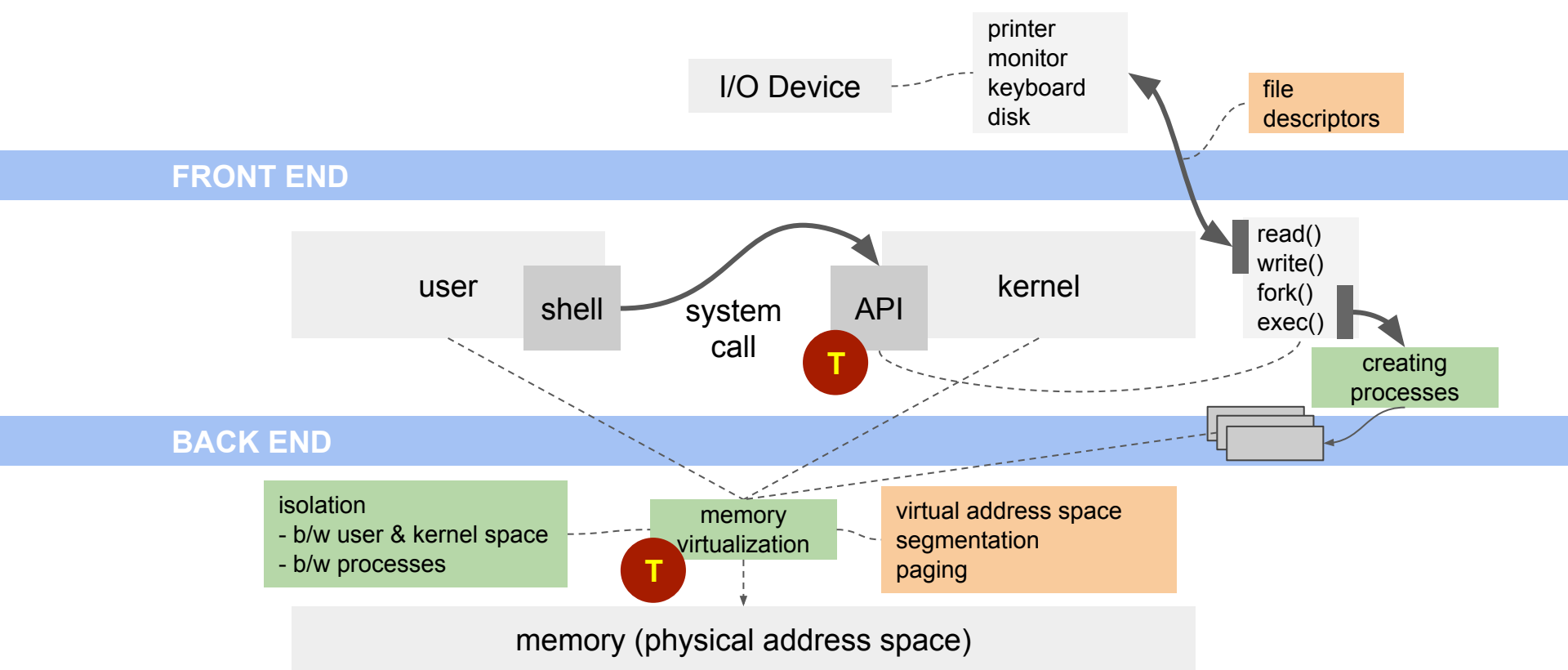




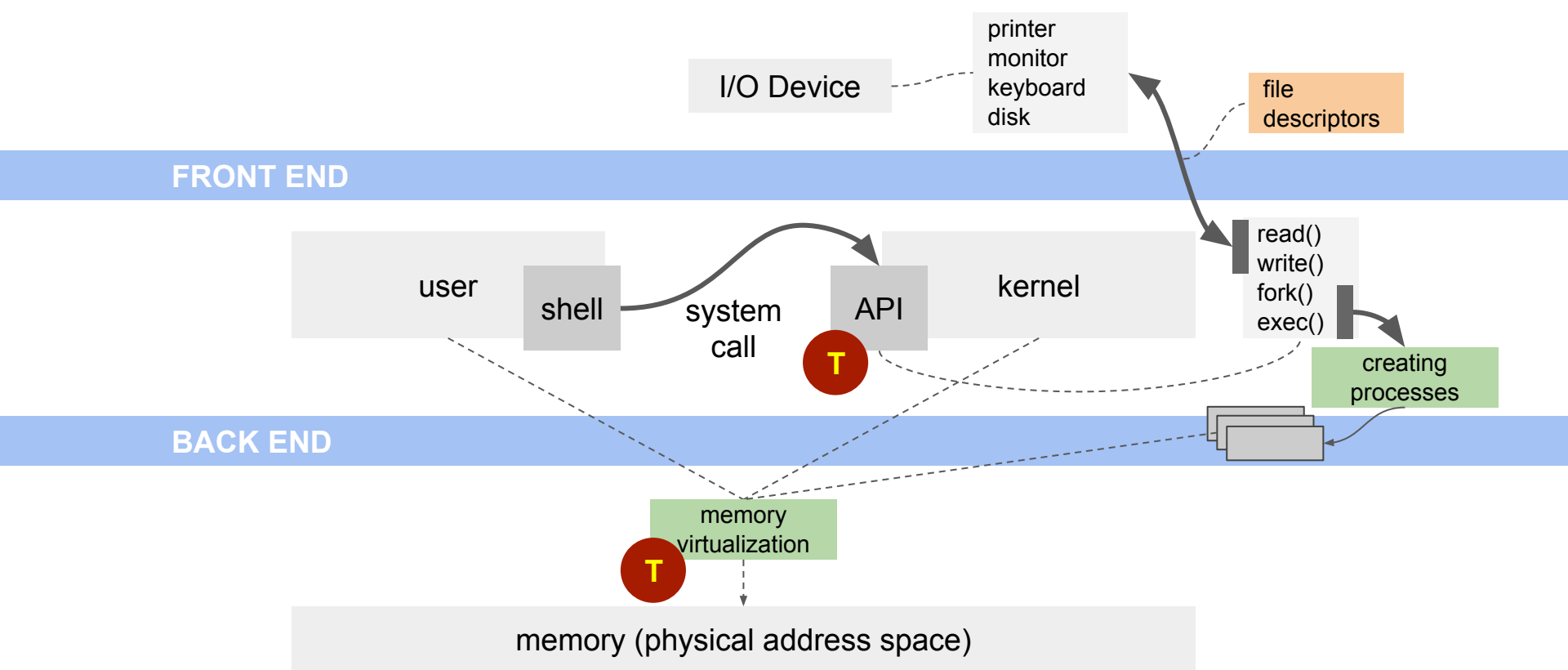


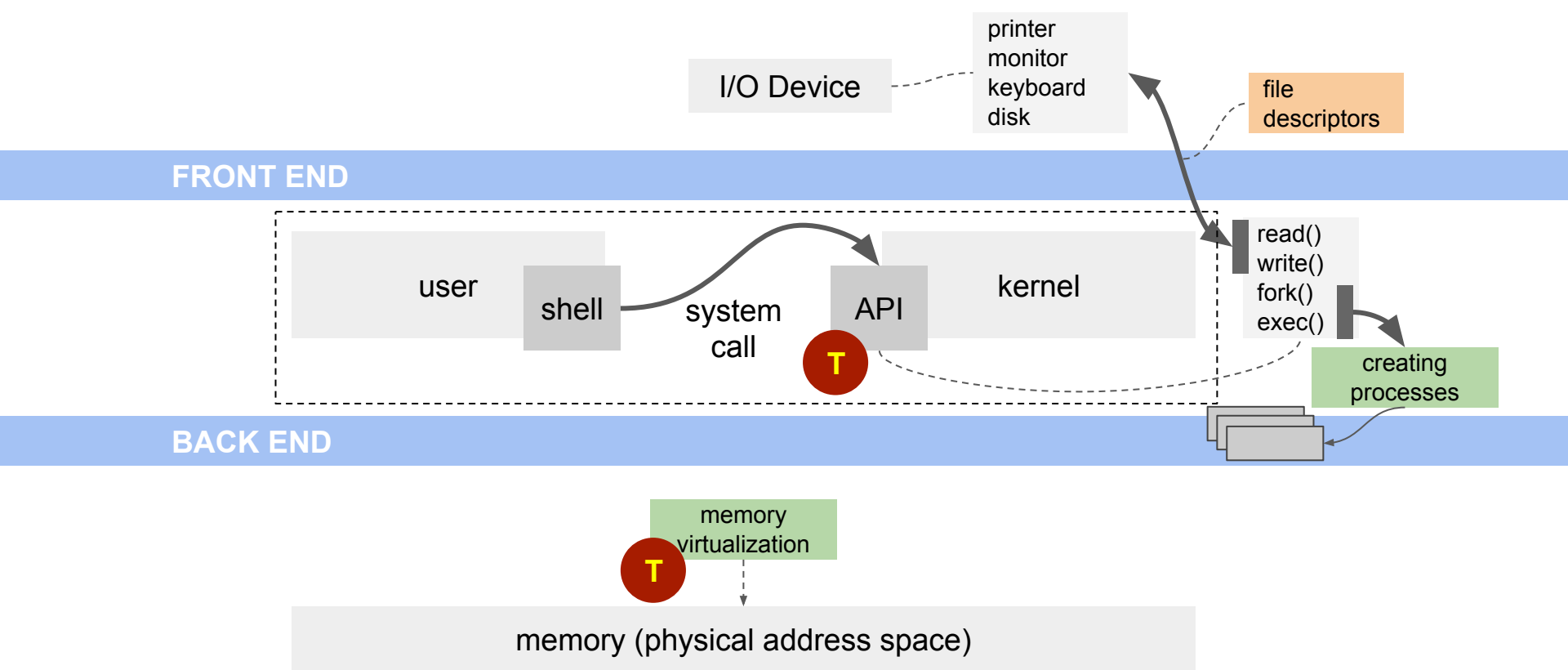


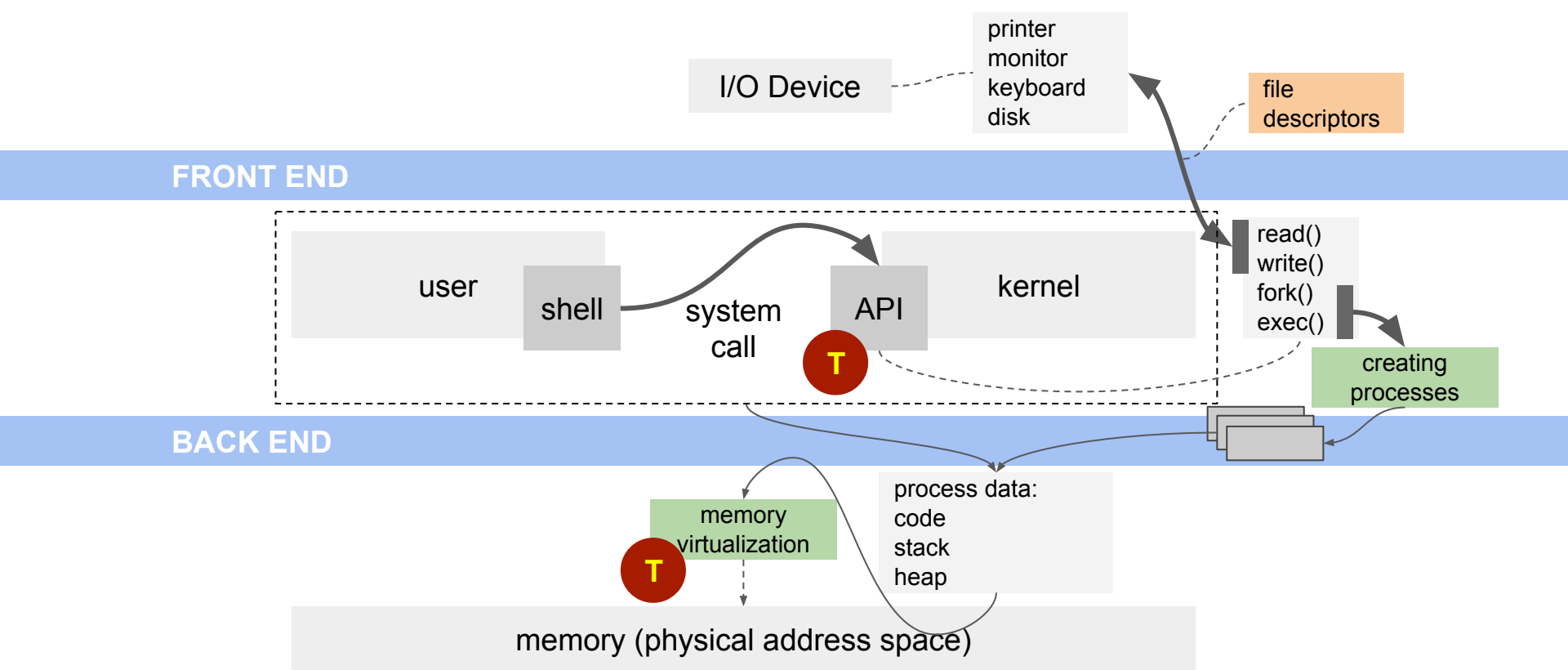


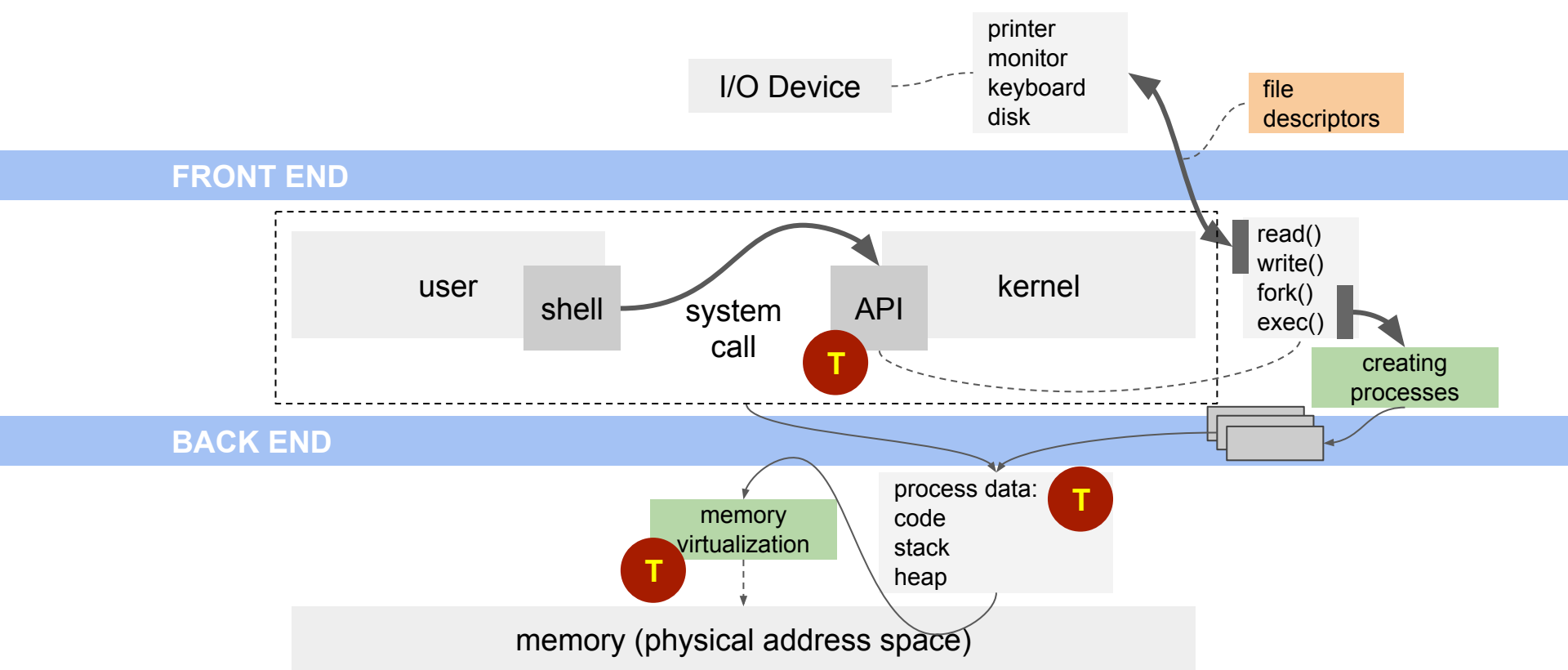


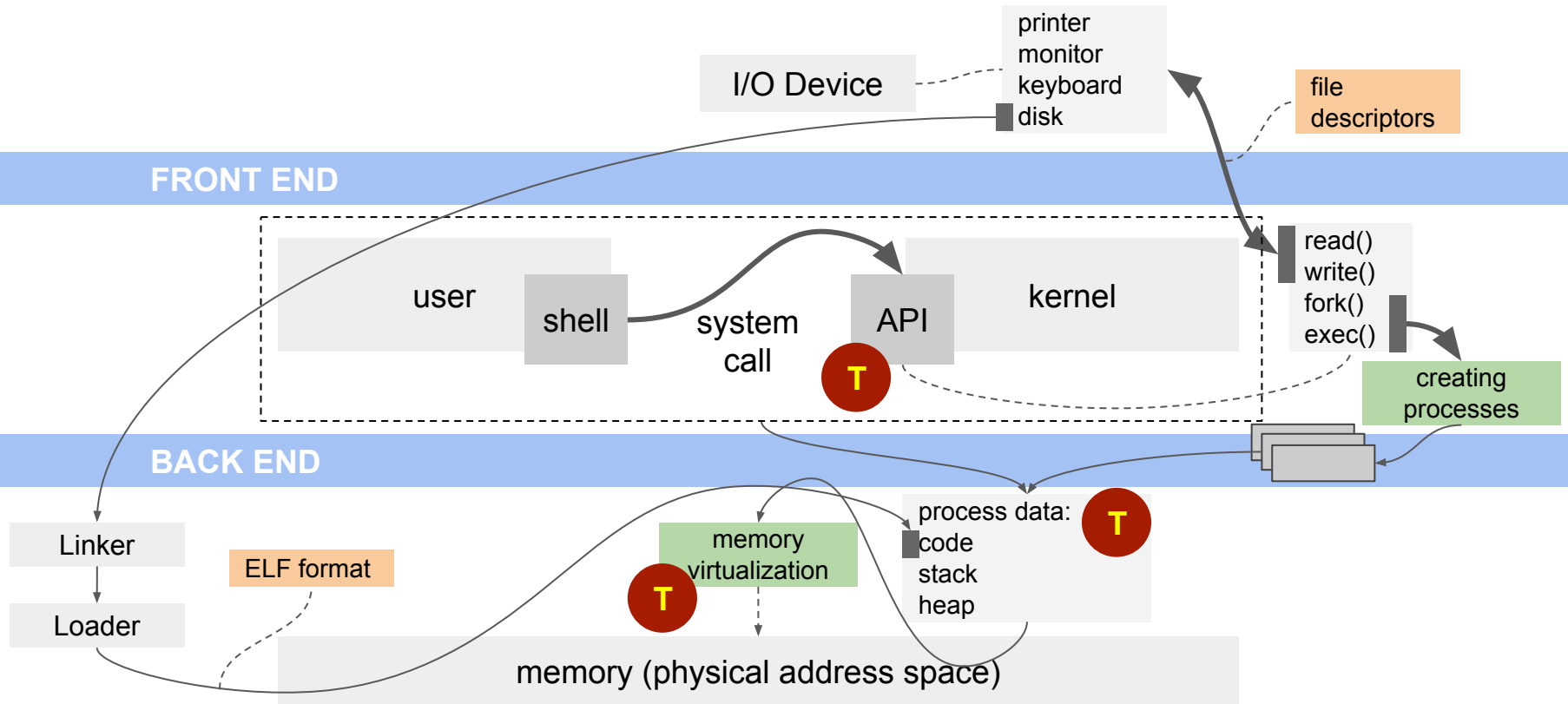
So what exactly must be in memory?

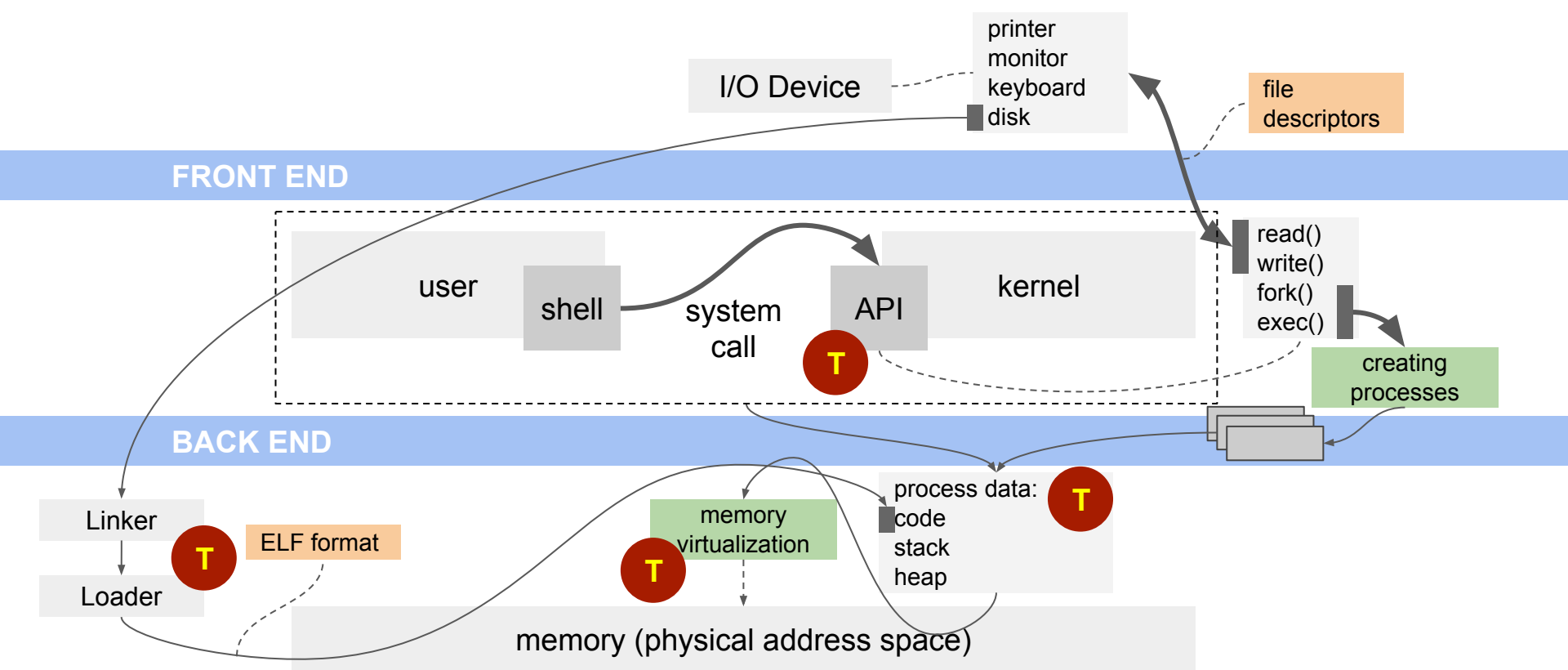




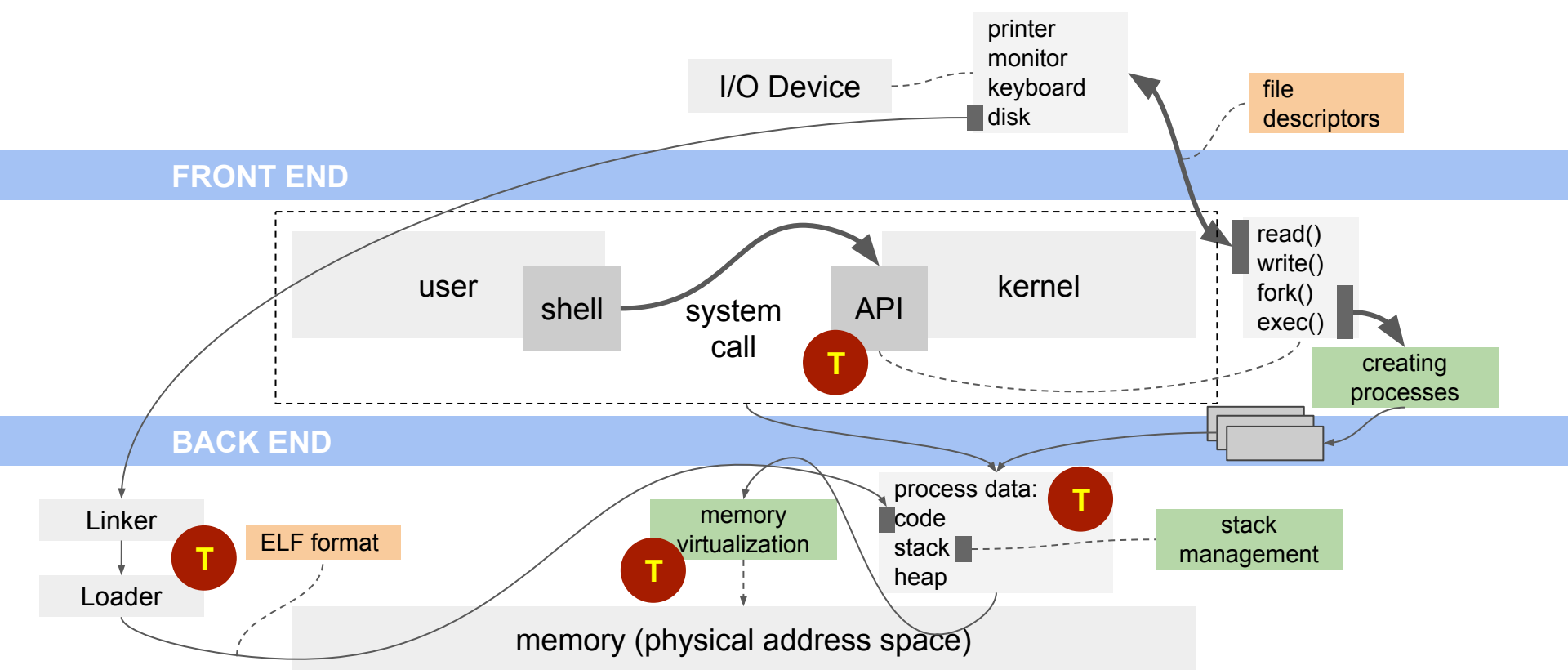


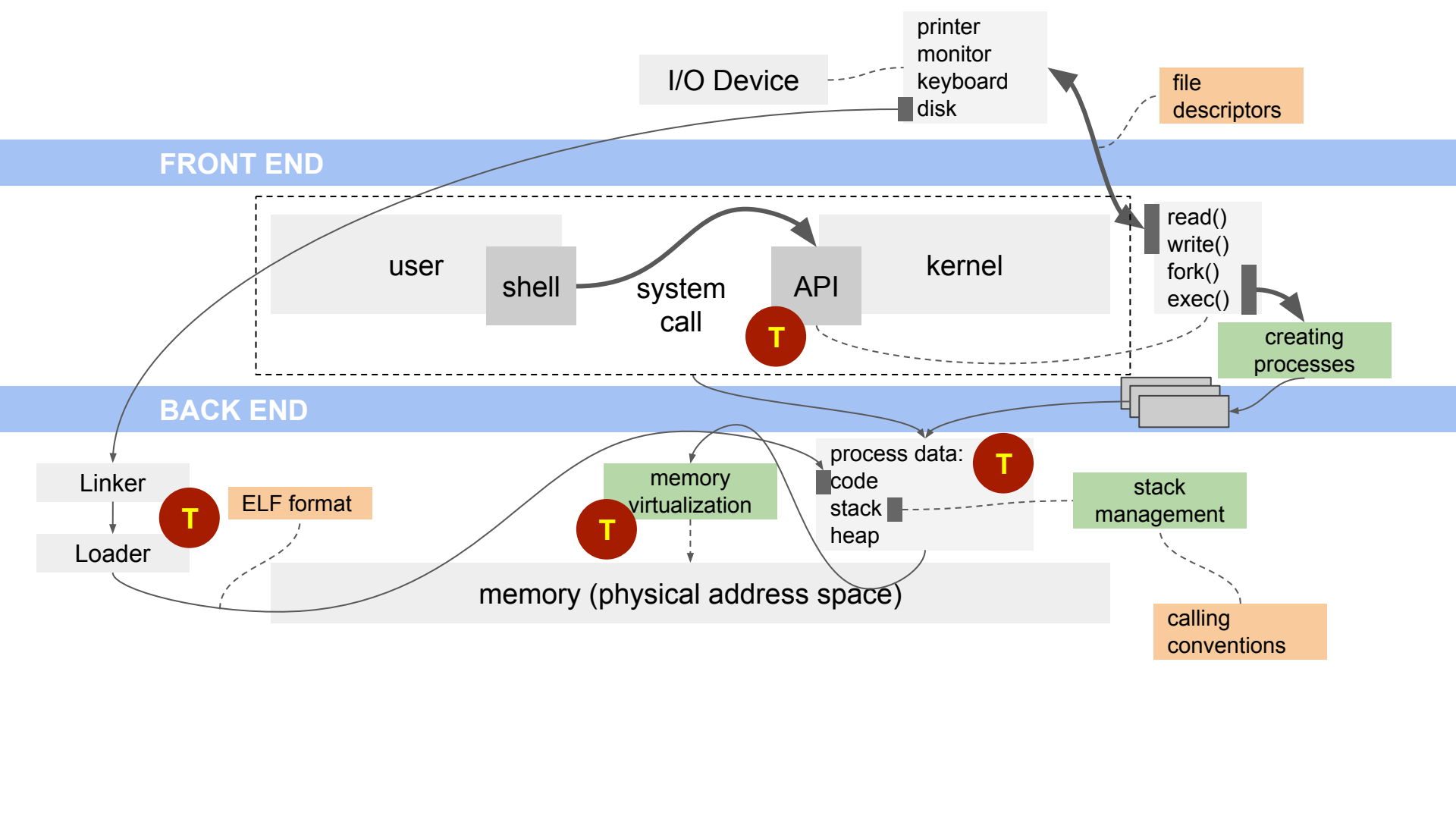


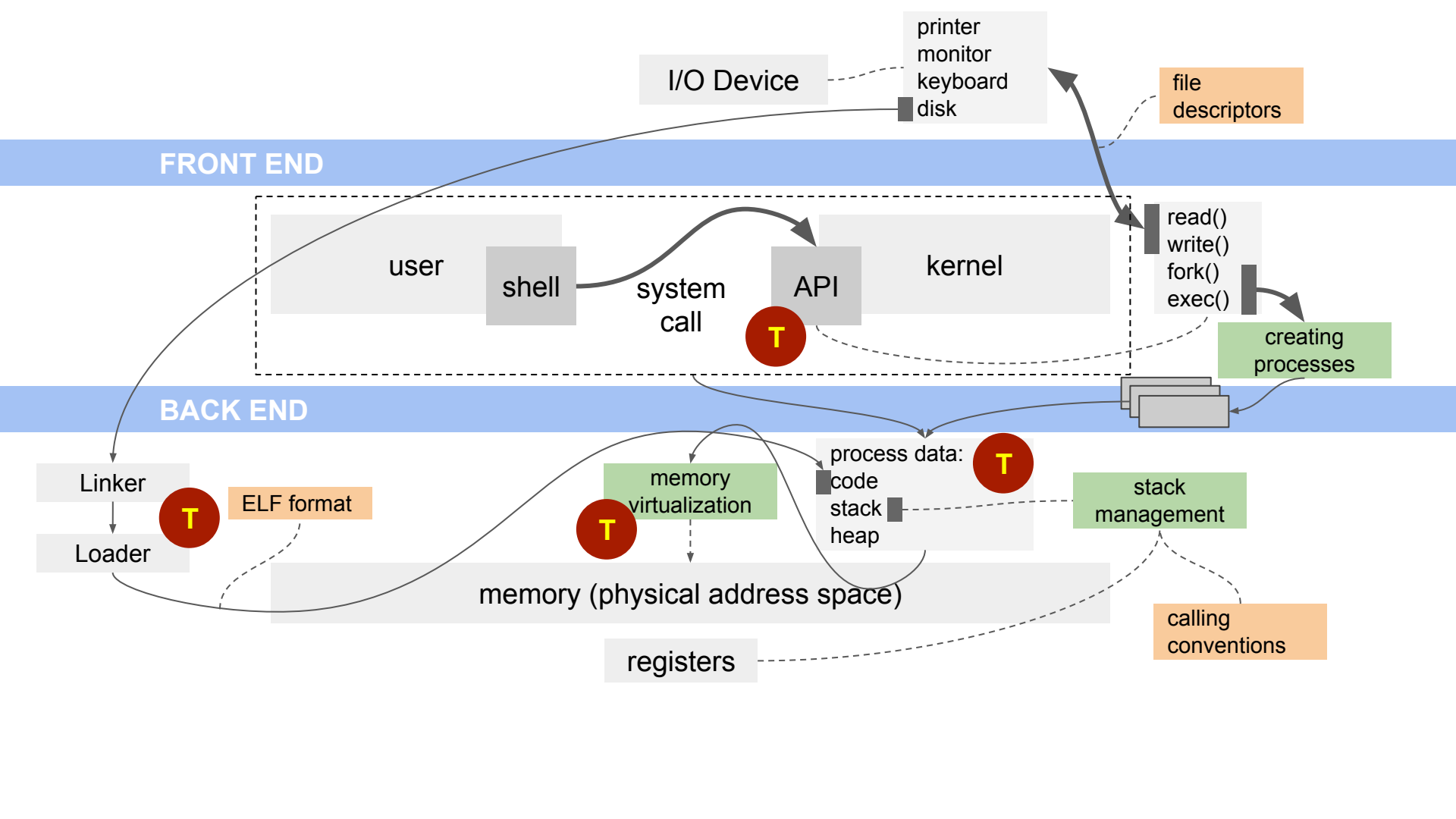


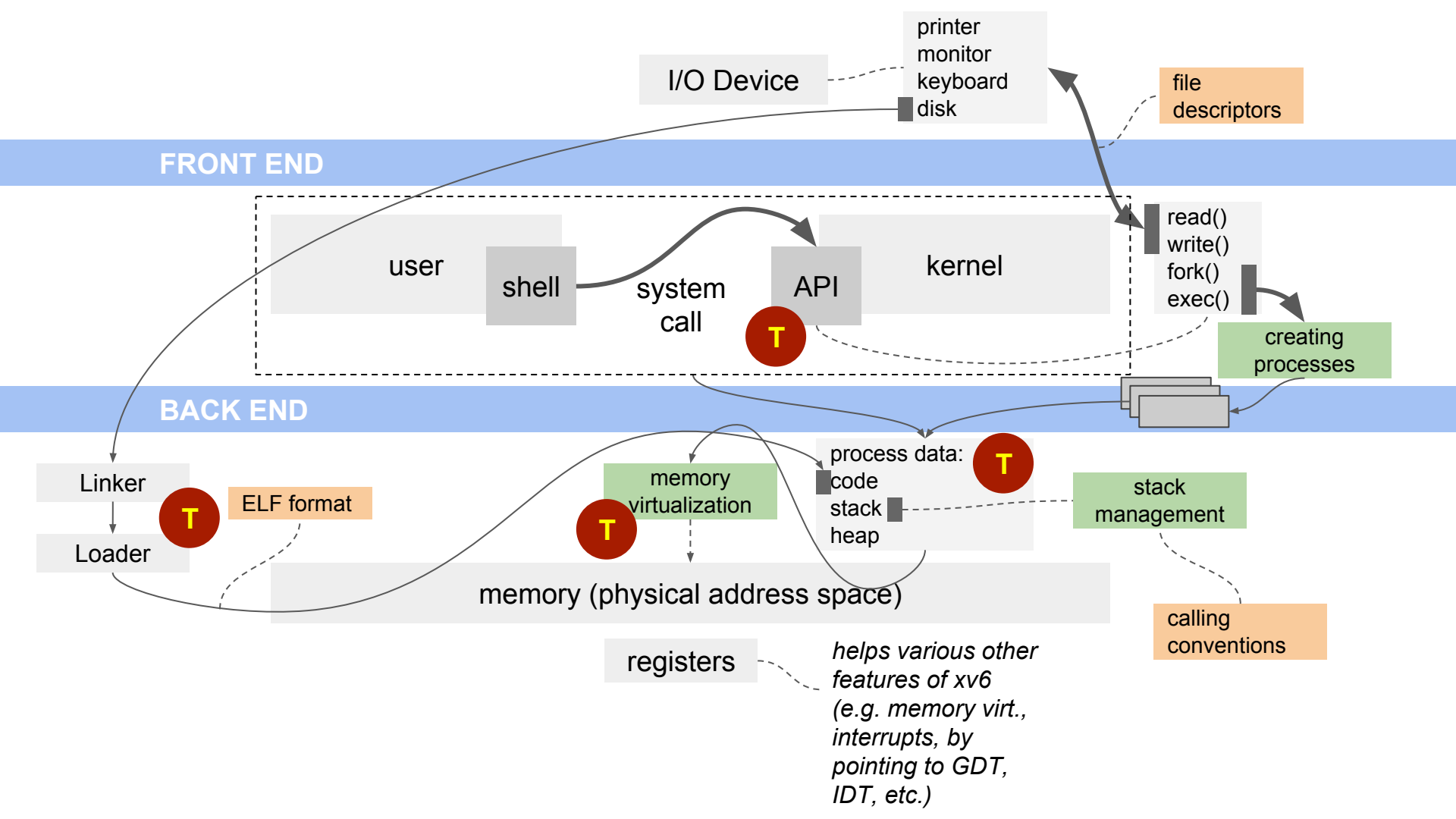


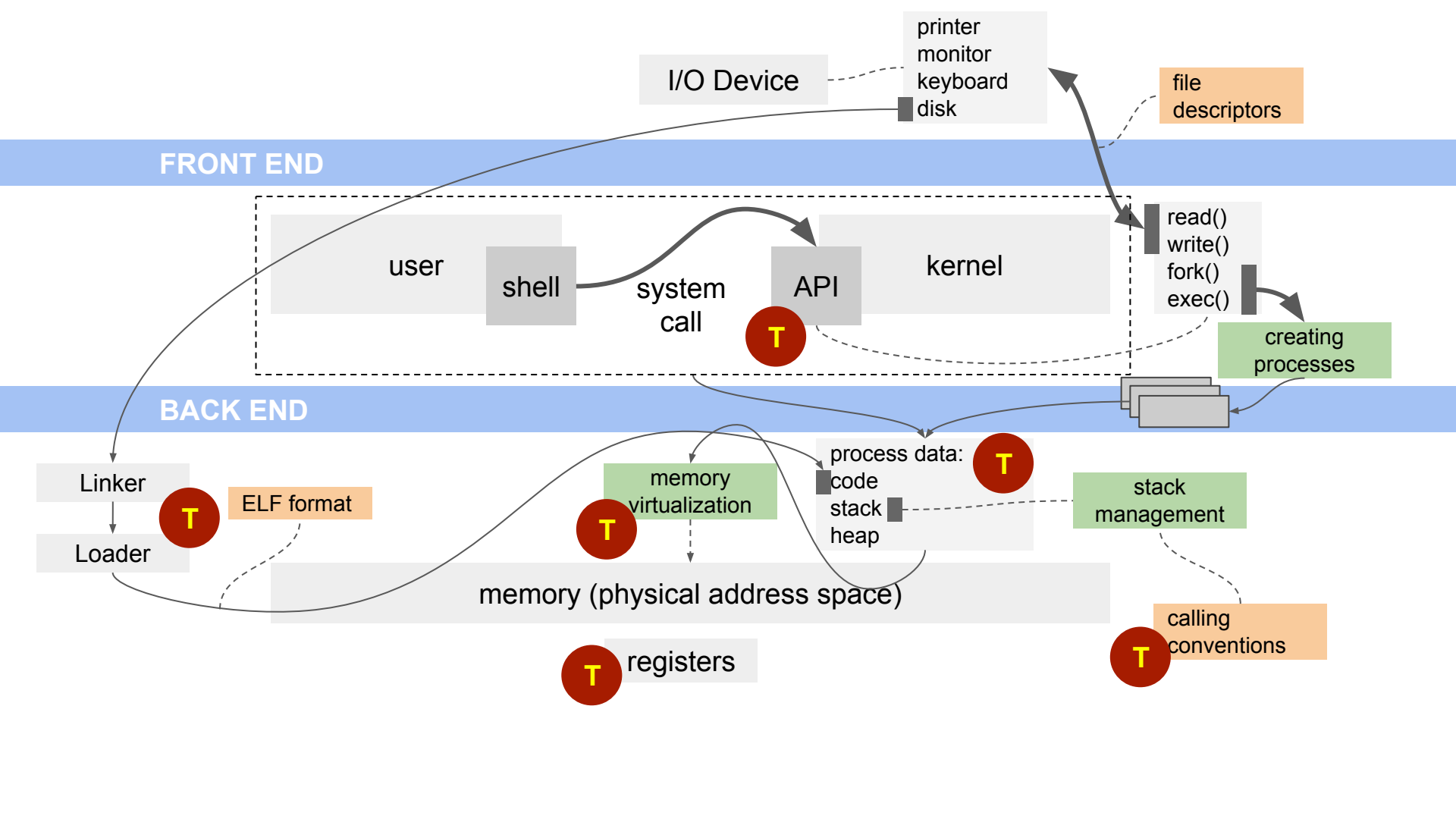
Need to handle dynamic info.



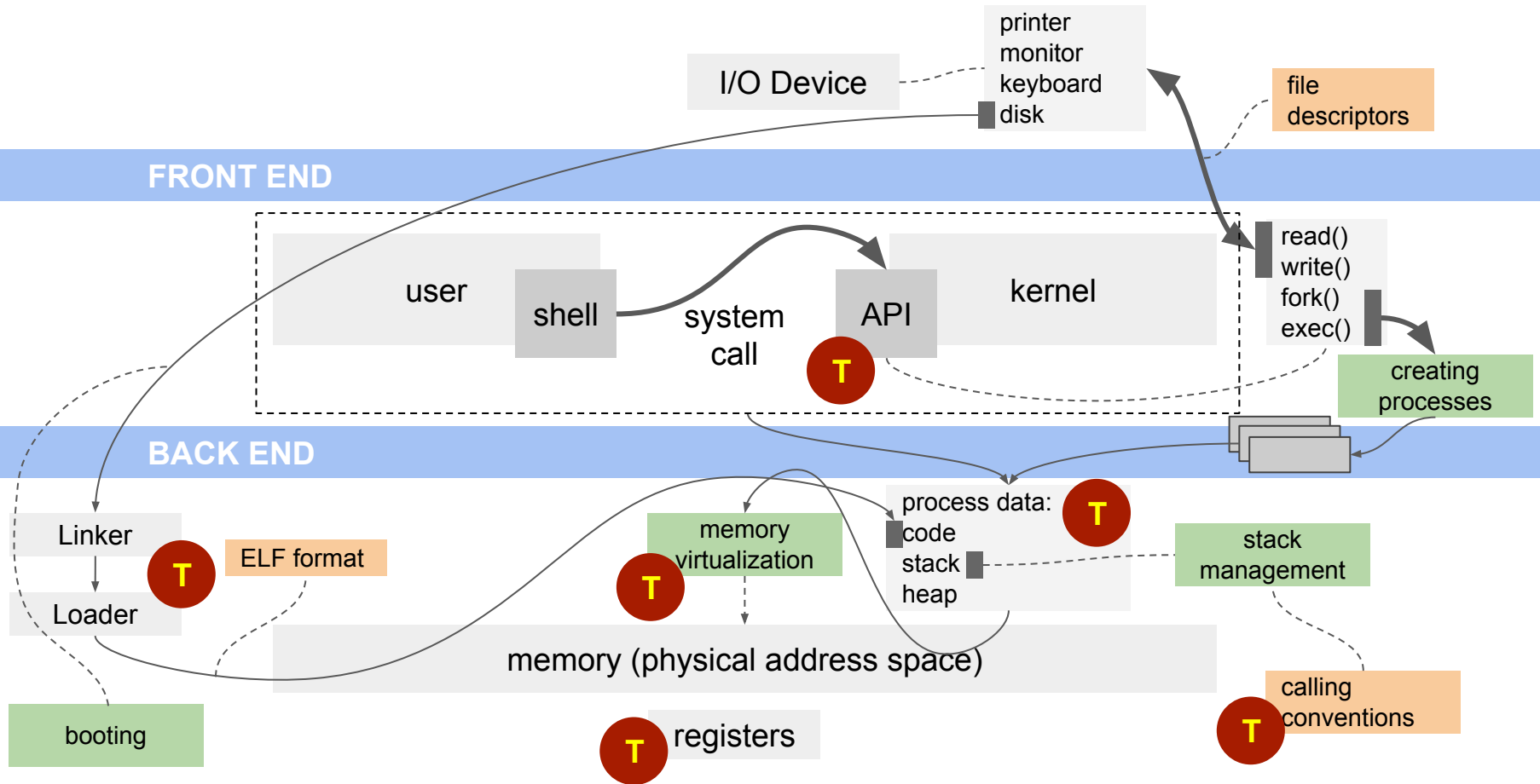


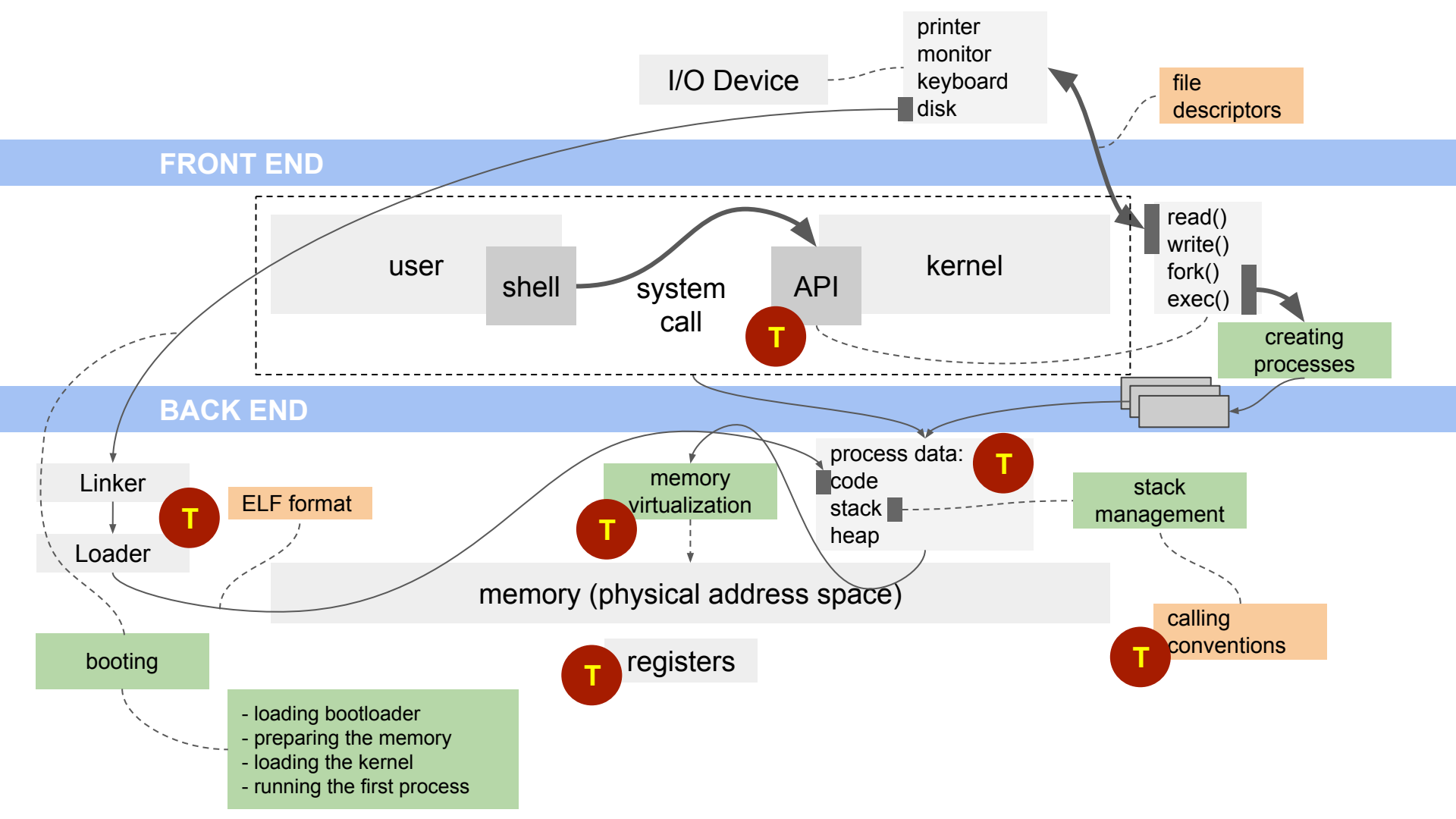


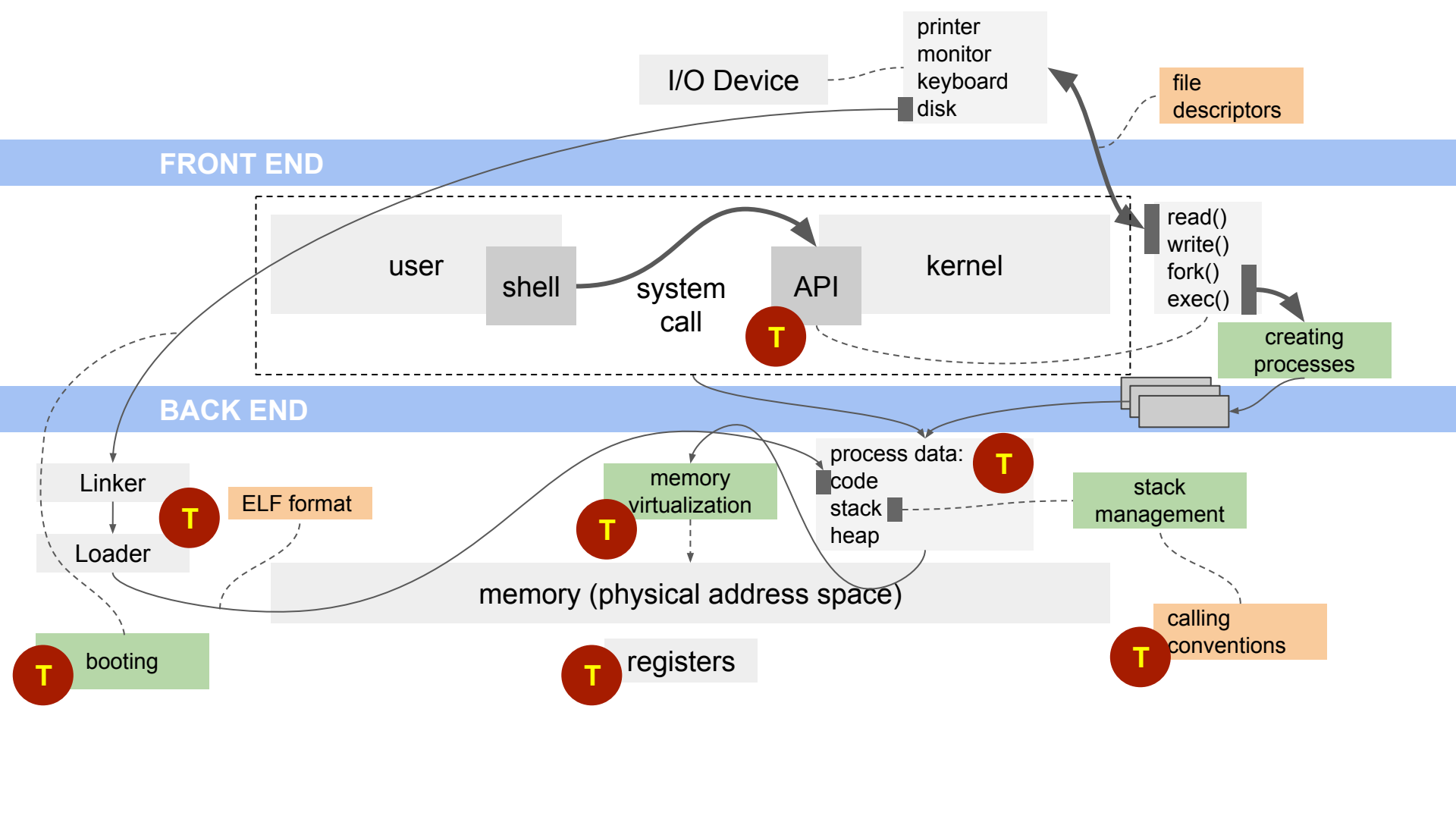




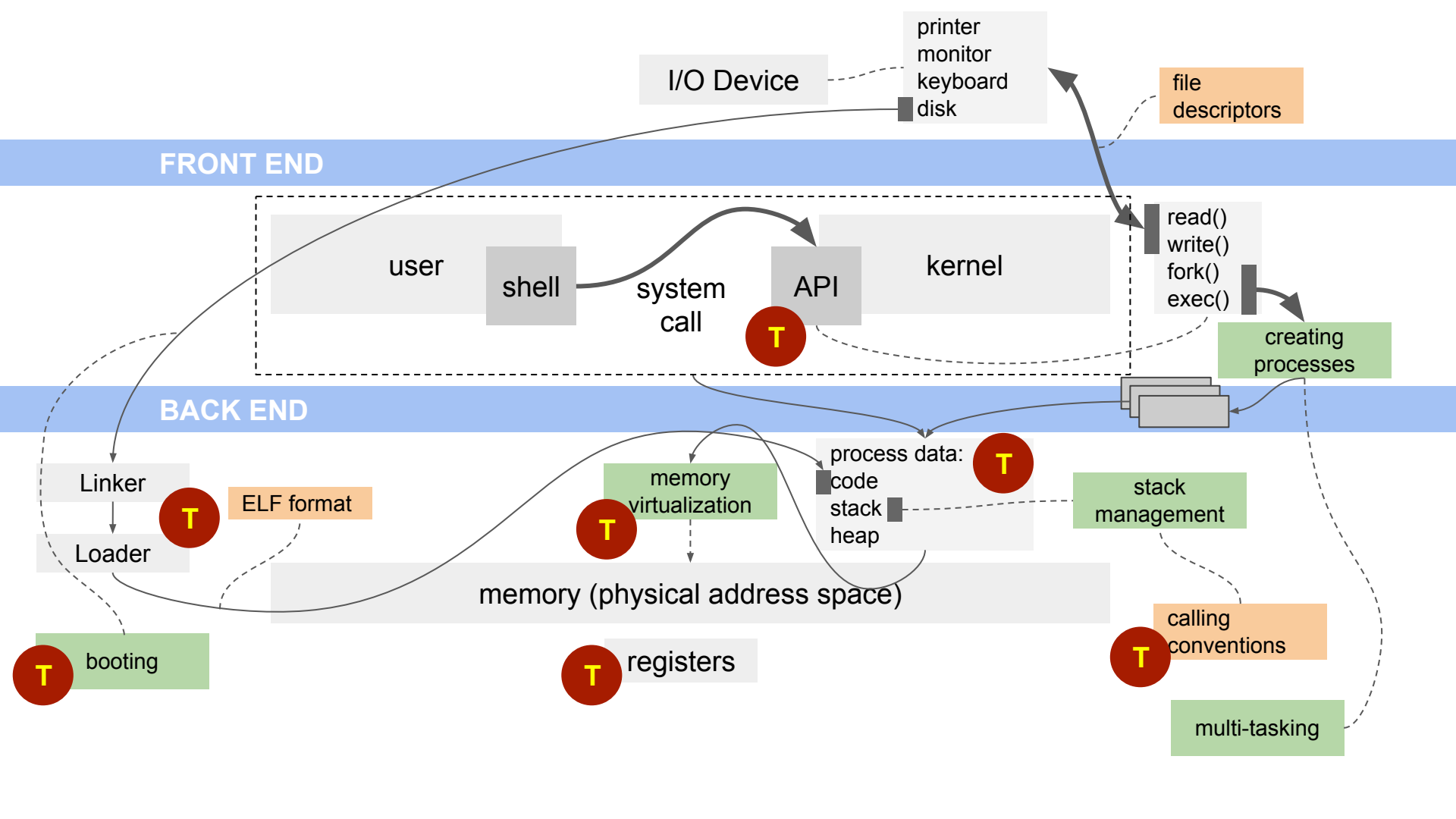
Did we forget to load the OS -- or
kernel -- itself?

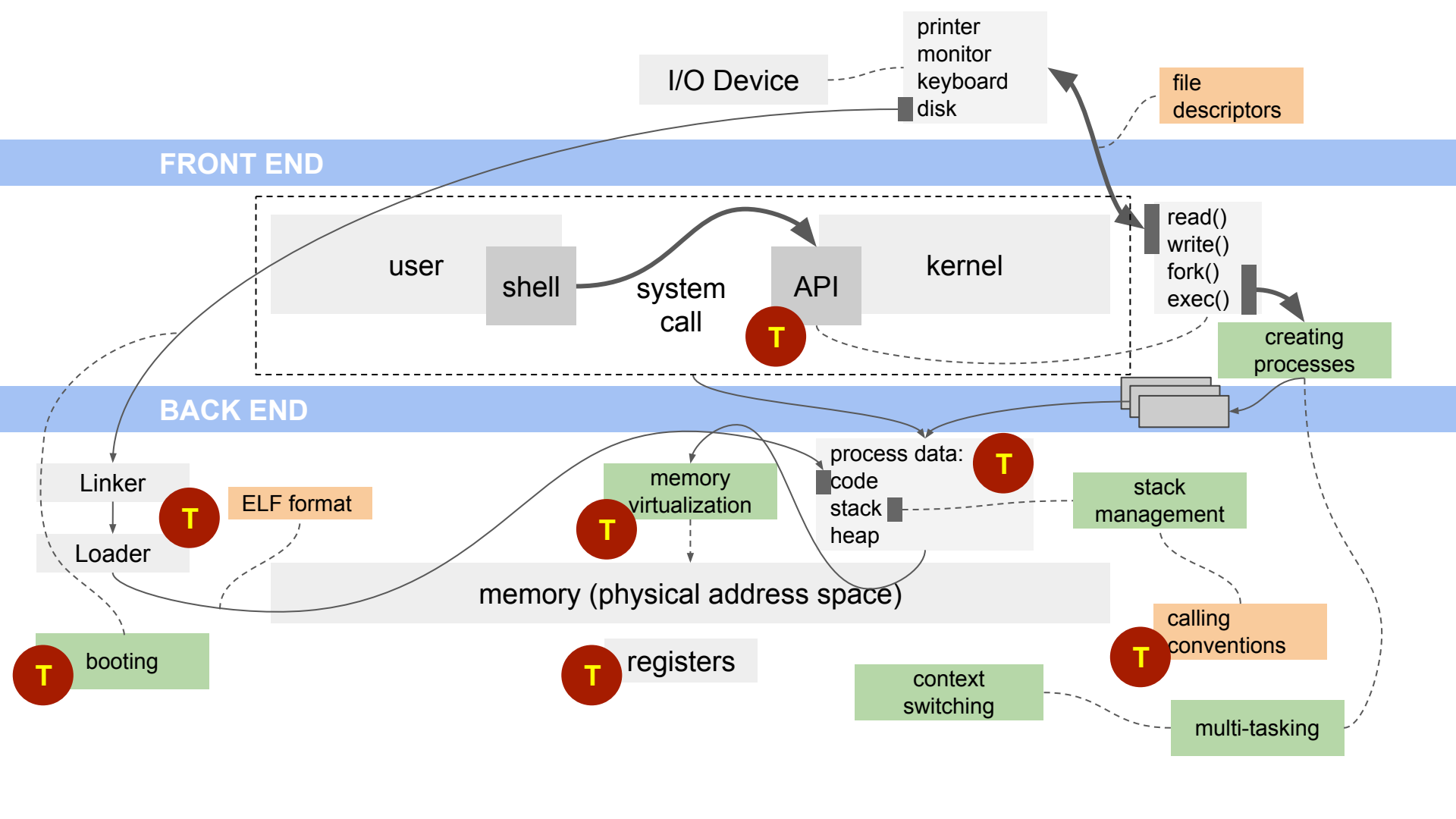


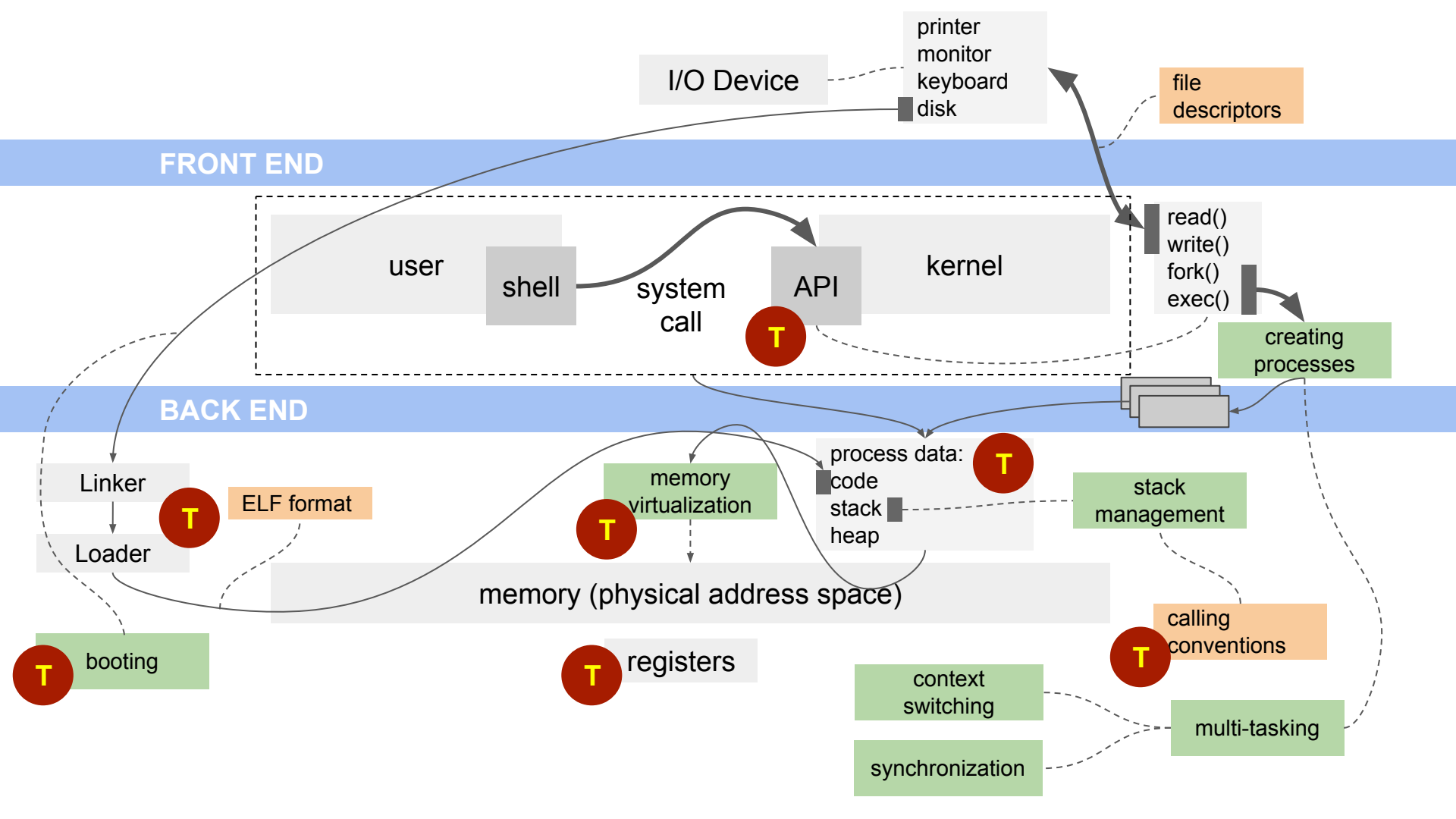


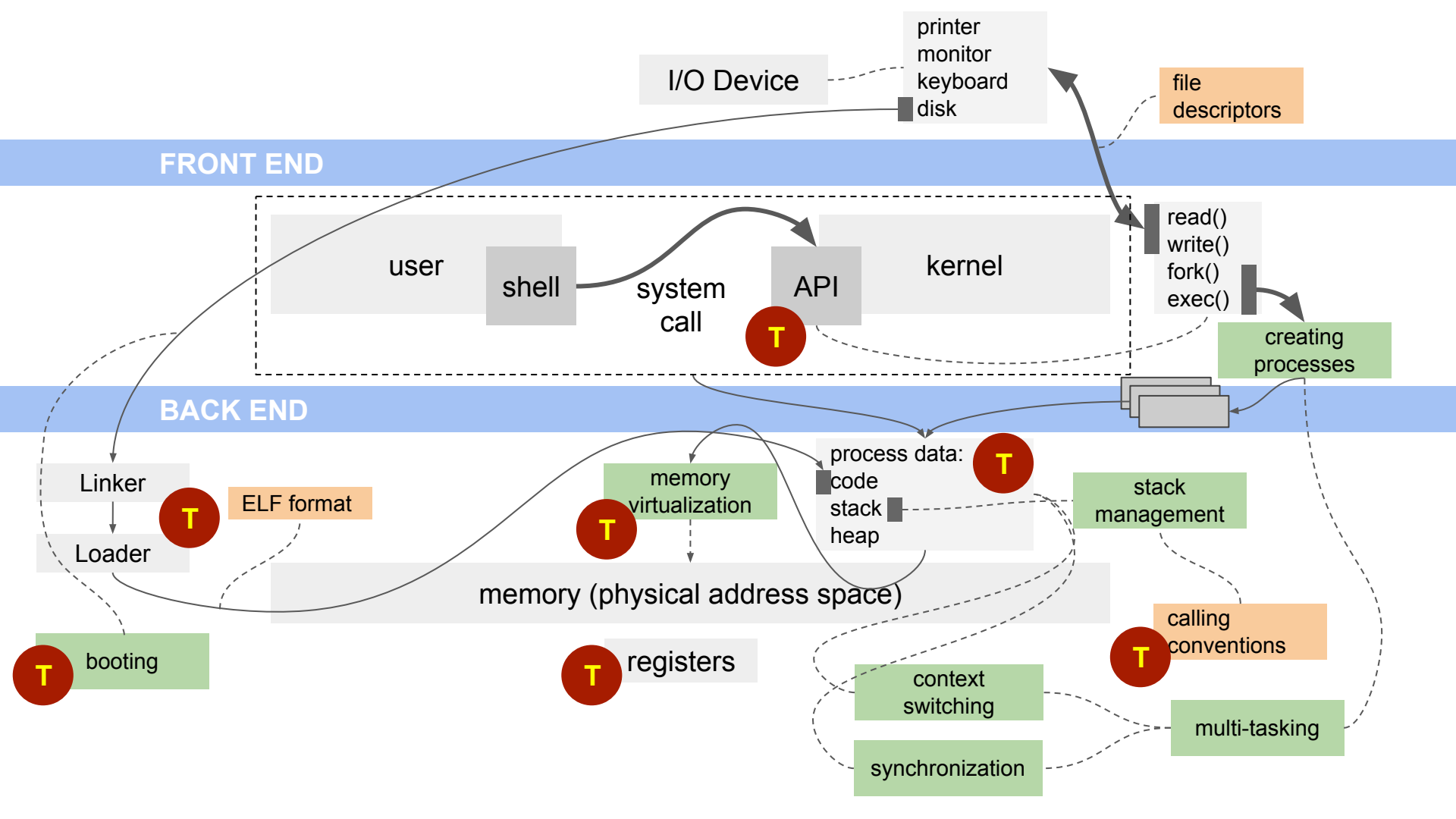


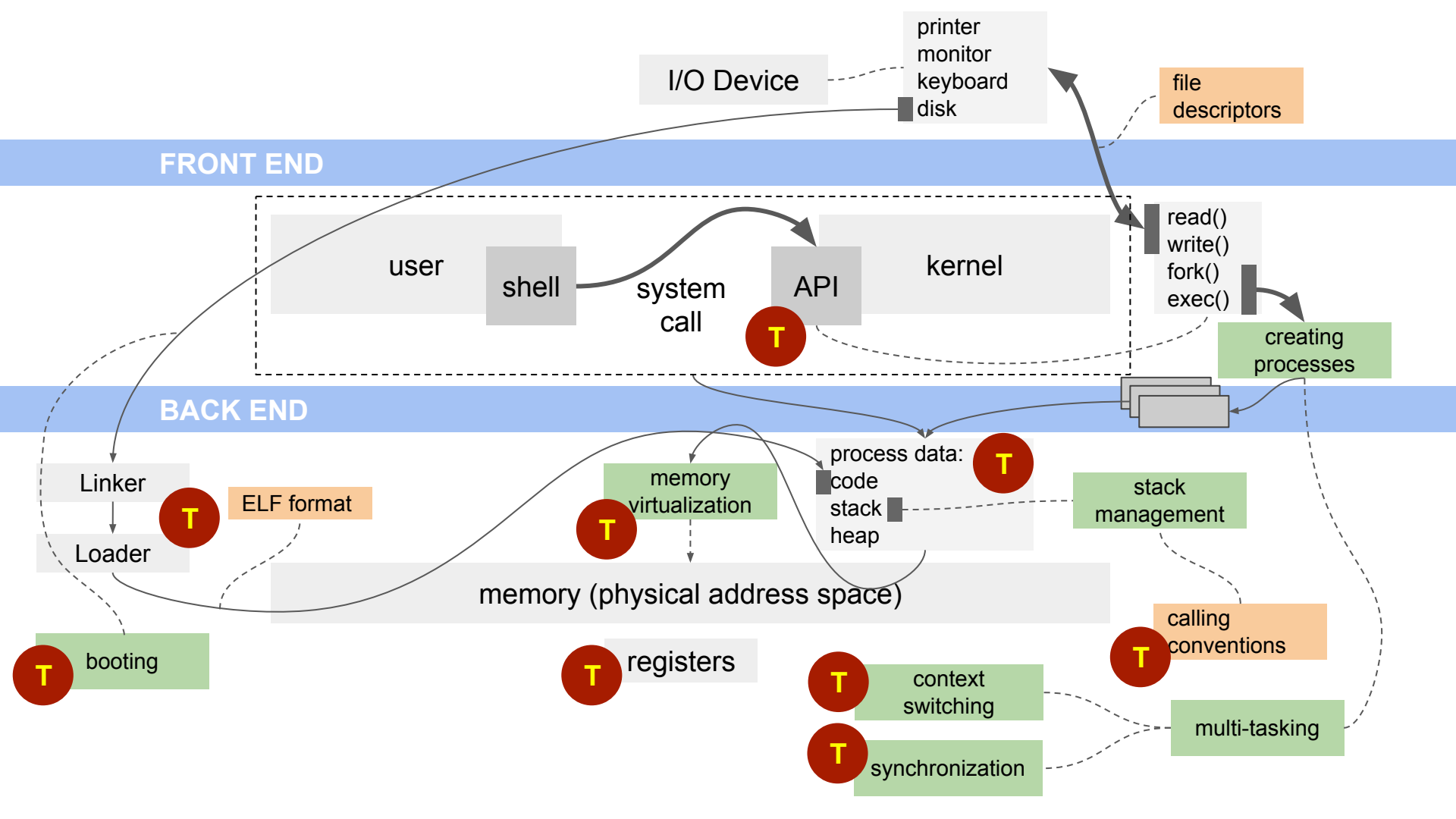
multi-tasking

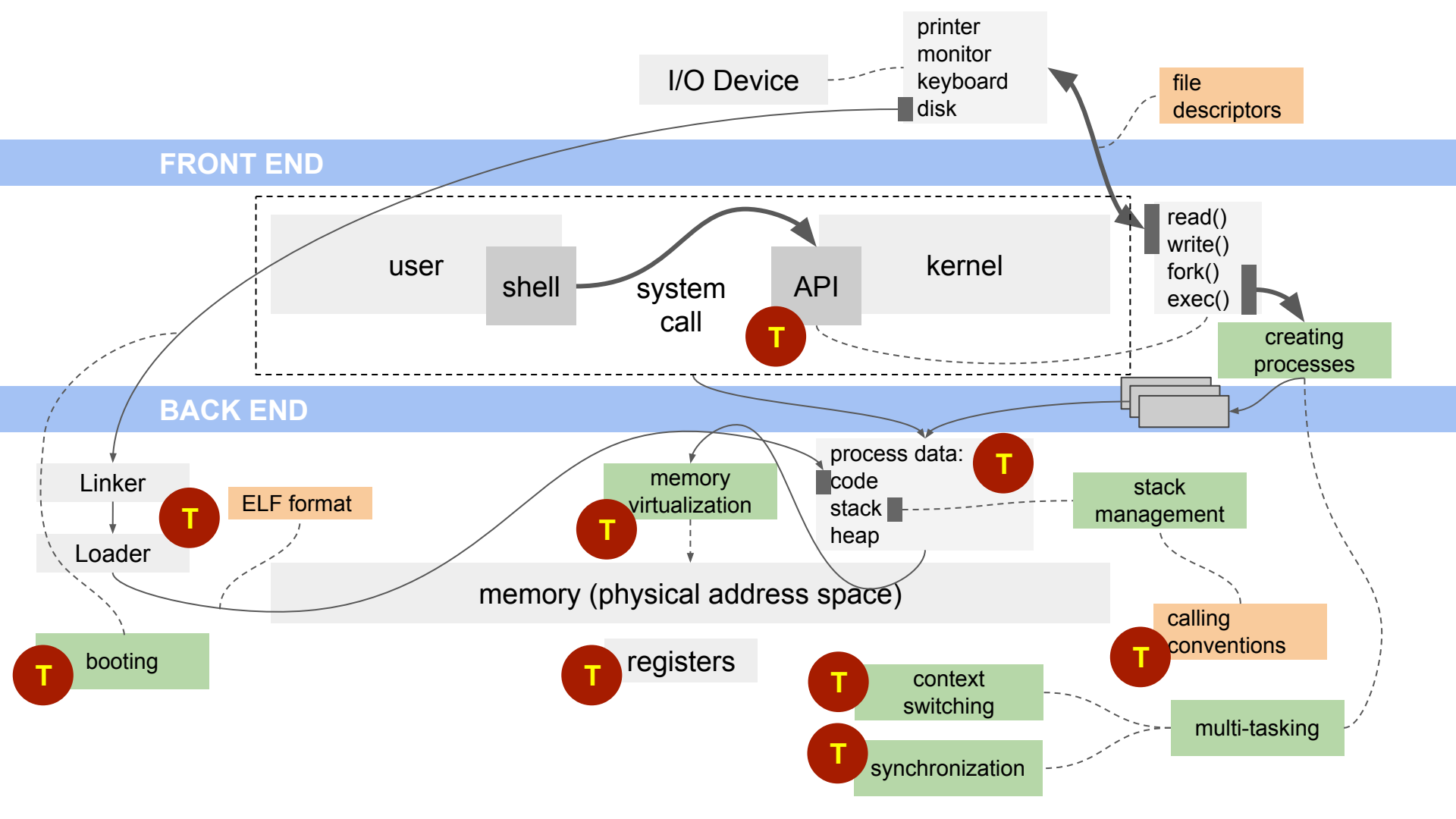


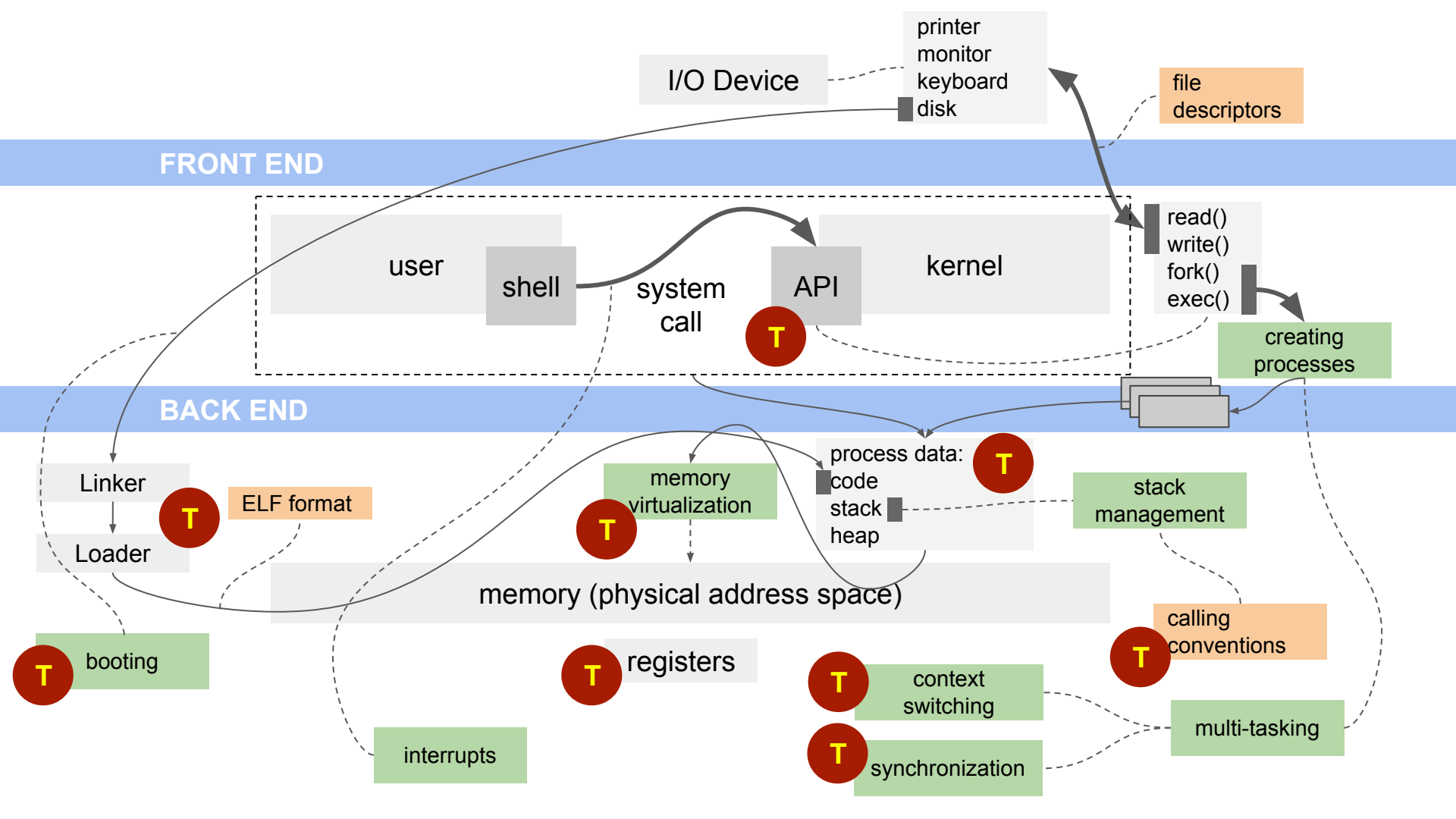


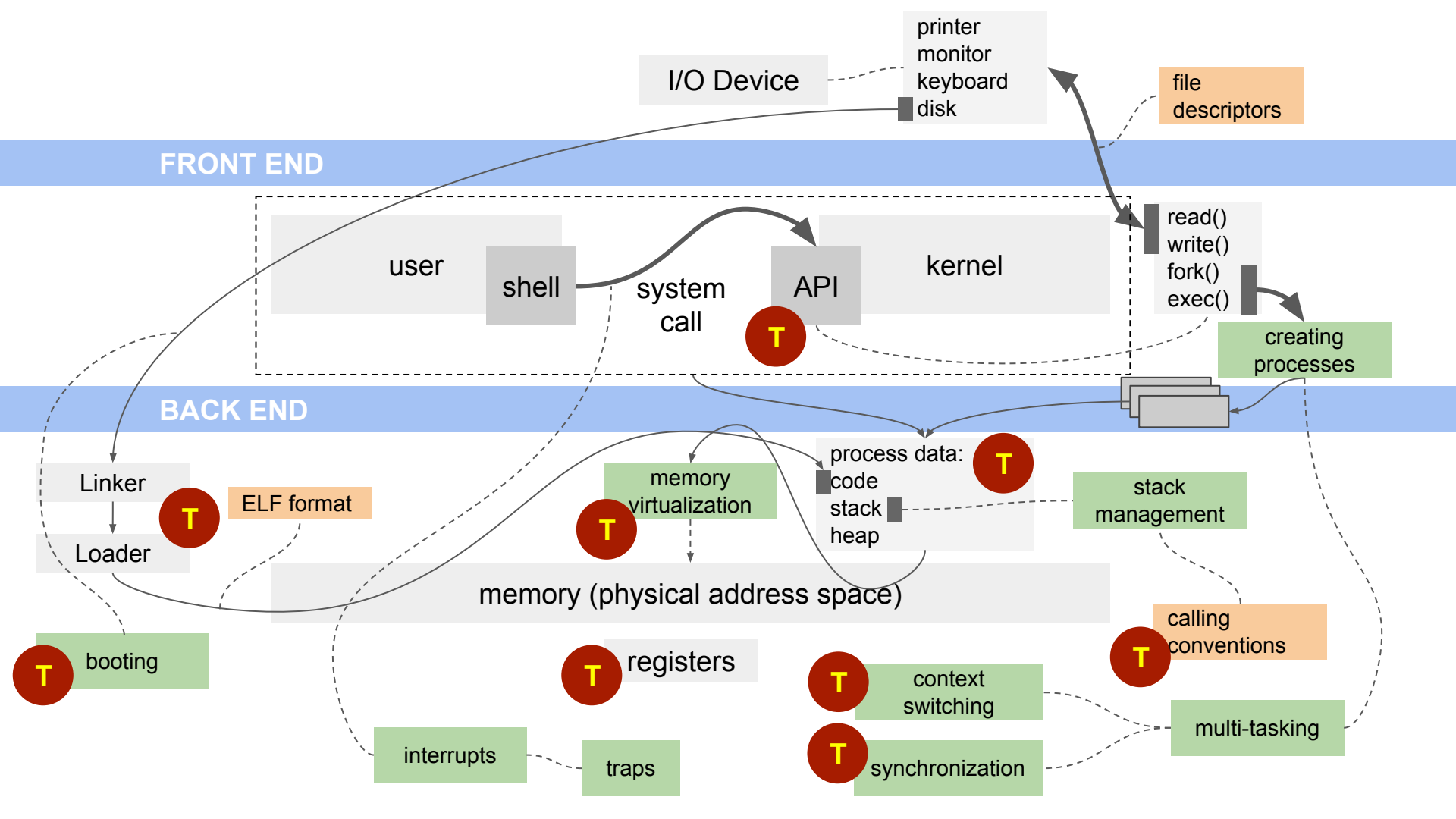


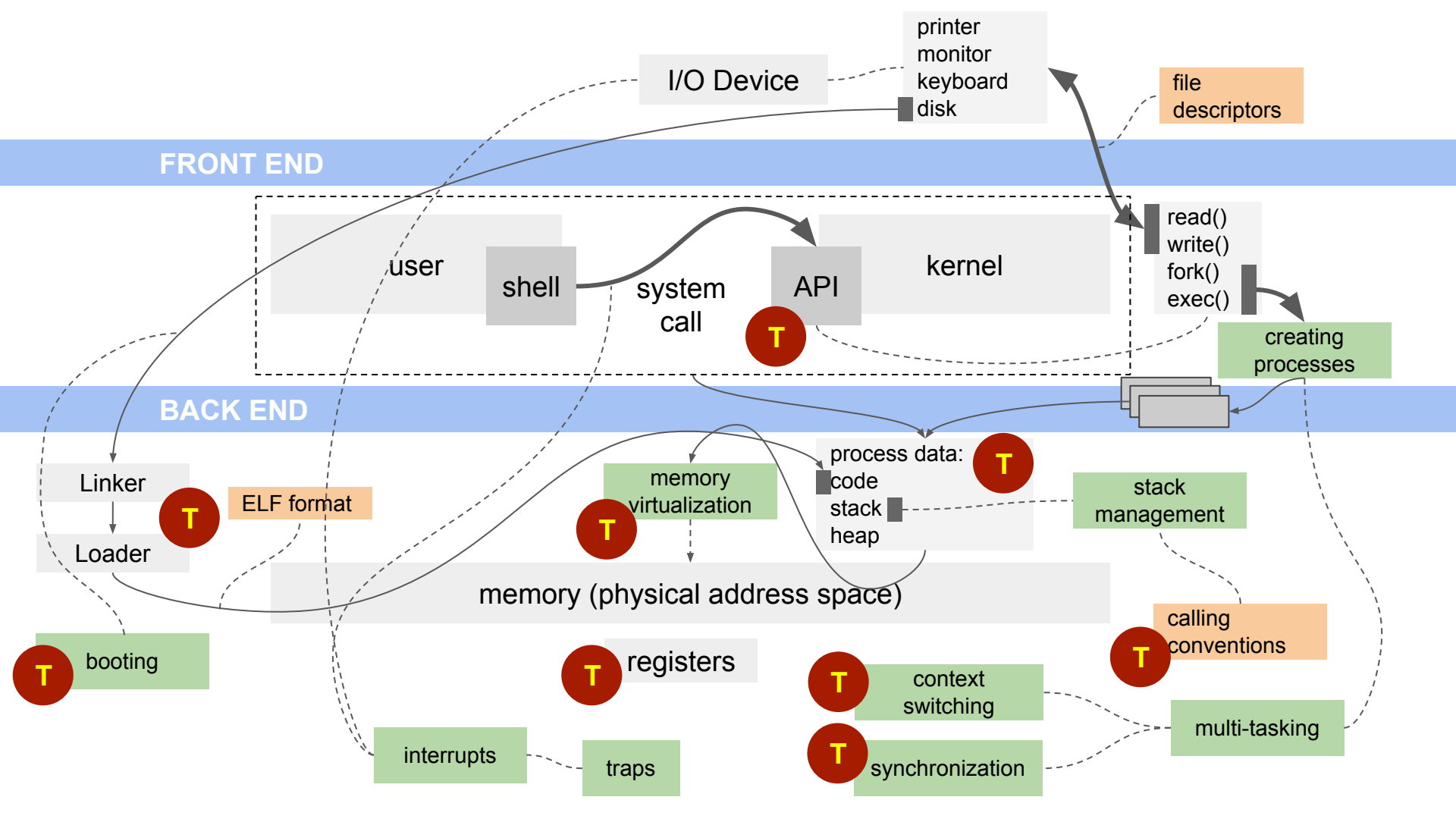


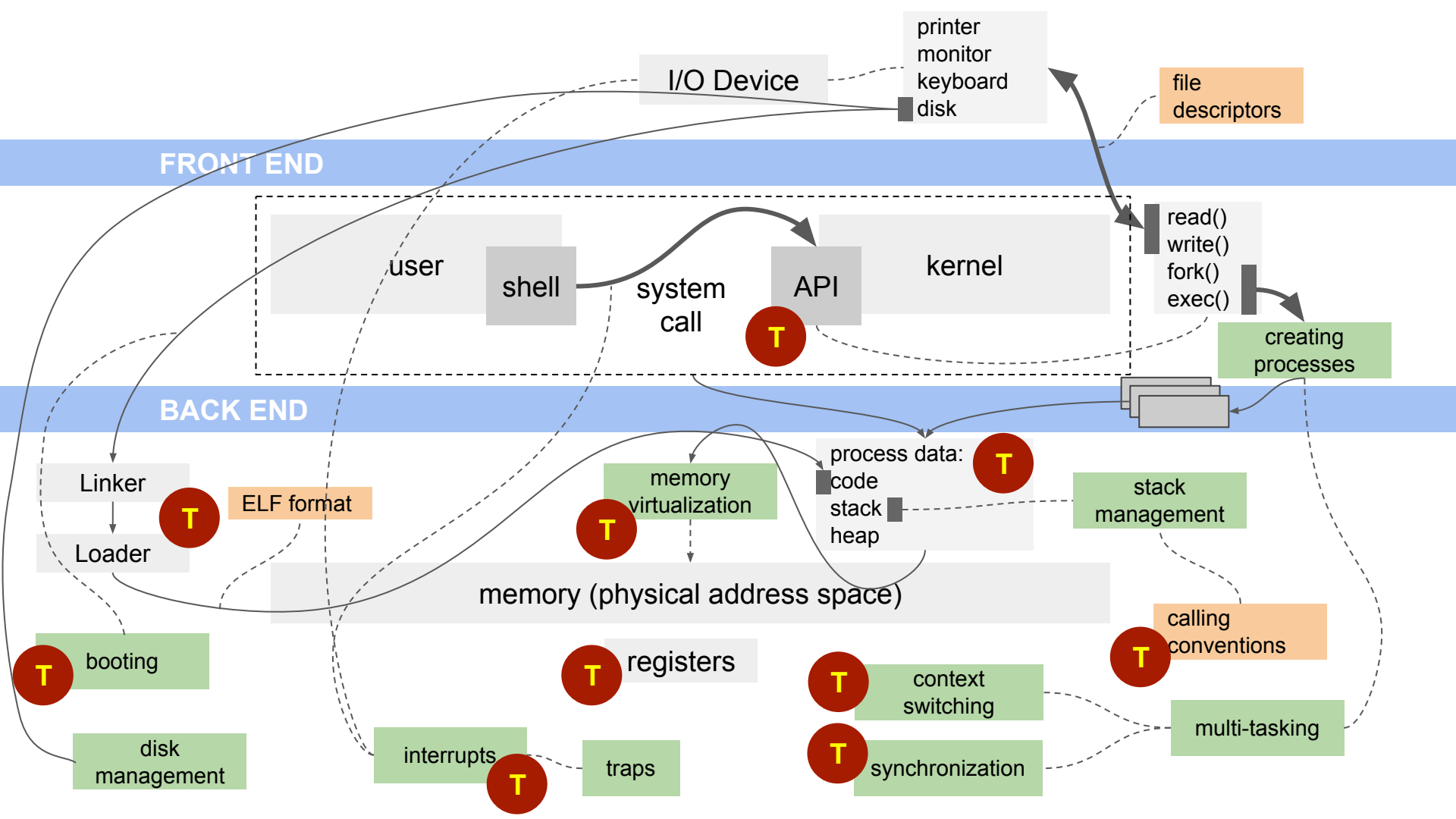


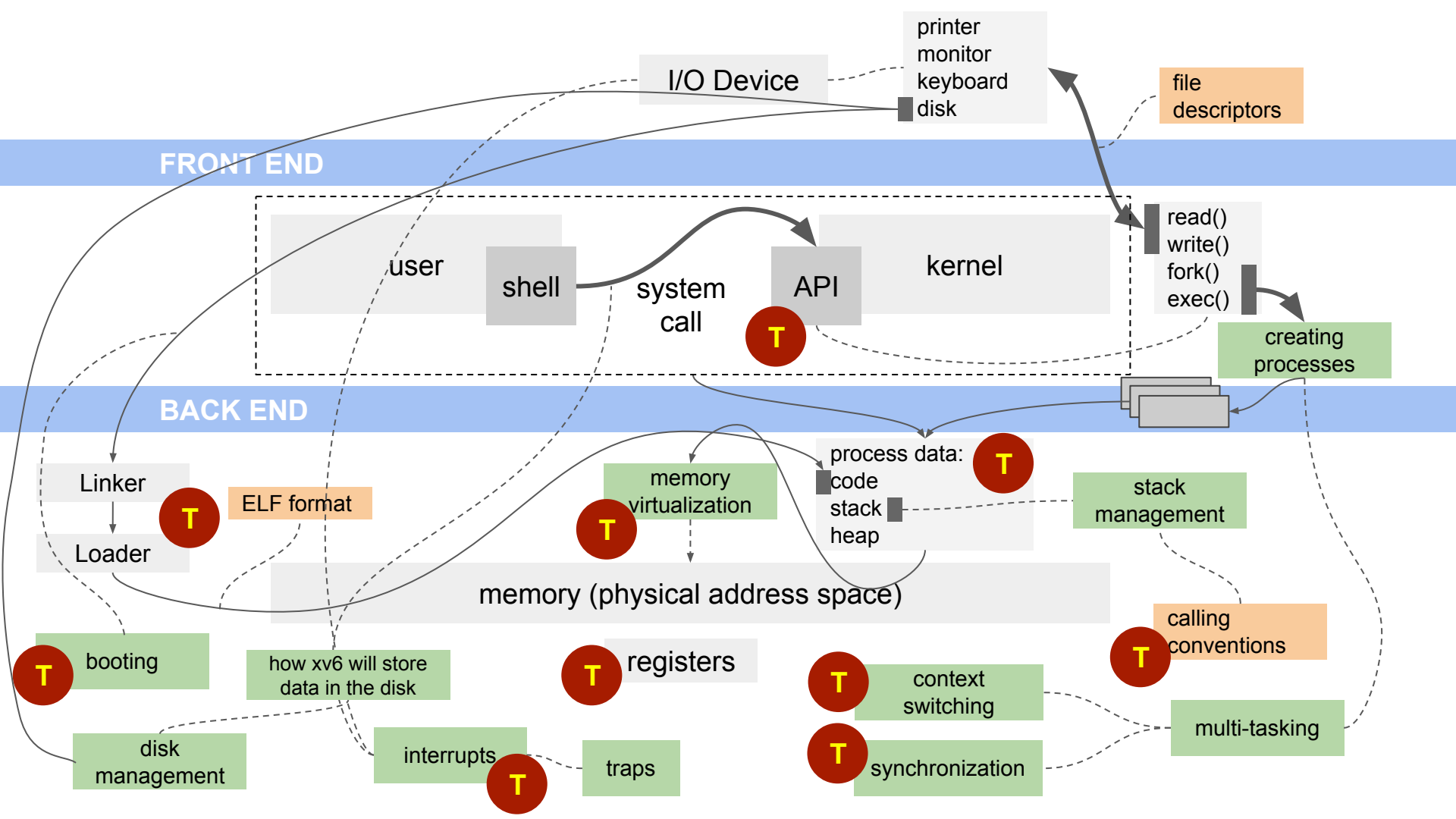


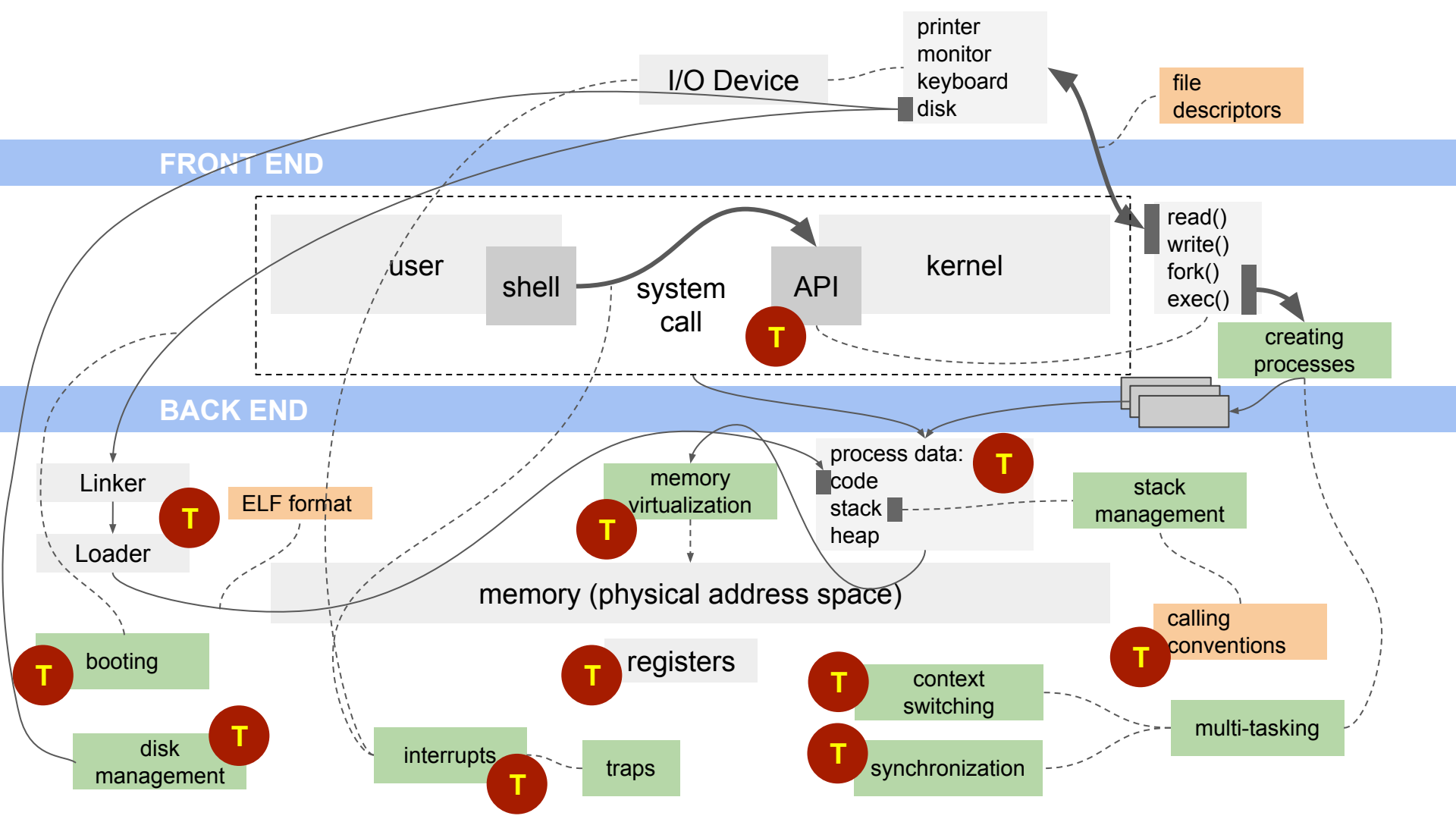






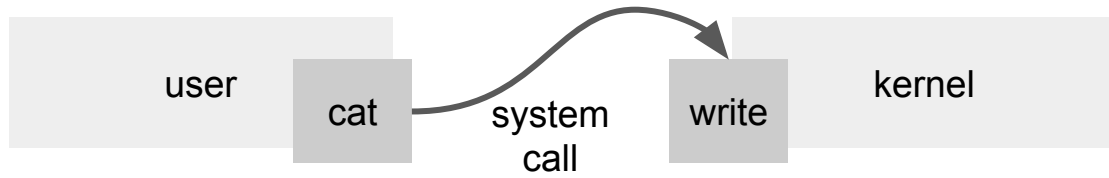






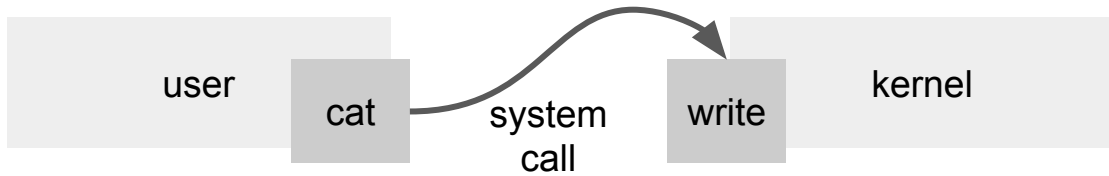
Now ... System Call Demo



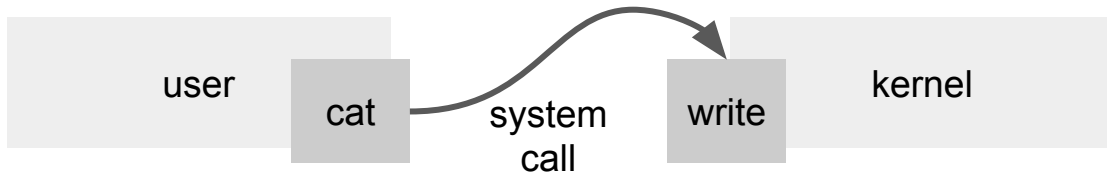


cat.c

```
if (write(1, buf, n) != n)
```



```
cat.c  if (write(1, buf, n) != n)
      defined here
usys.S  SYSCALL(write)
```



cat.c

```
if (write(1, buf, n) != n)
```

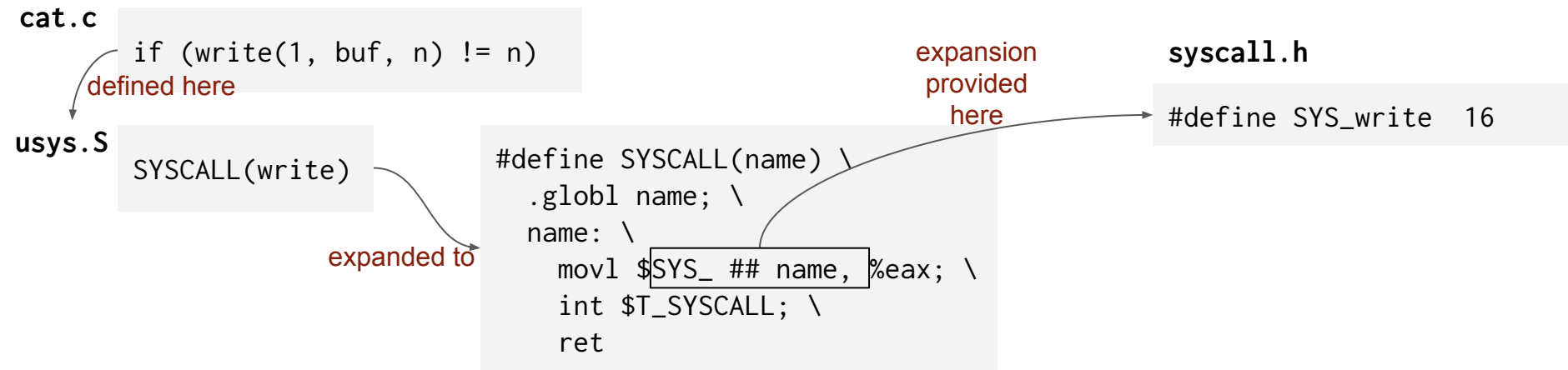
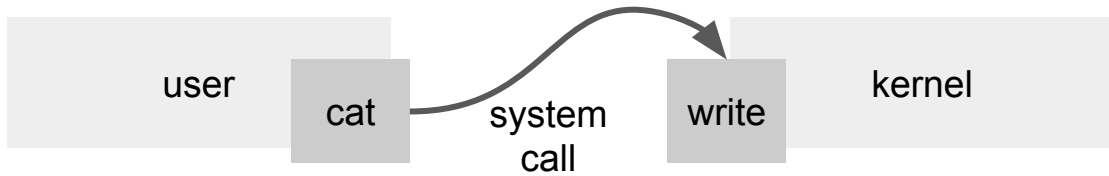
defined here

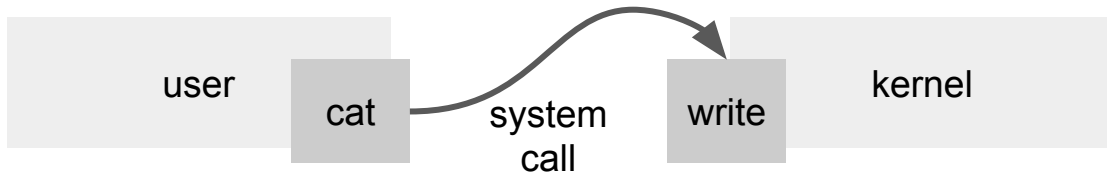
usys.S

```
SYSCALL(write)
```

expanded to

```
#define SYSCALL(name) \  
    .globl name; \  
    name: \  
        movl $SYS_ ## name, %eax; \  
        int $T_SYSCALL; \  
        ret
```

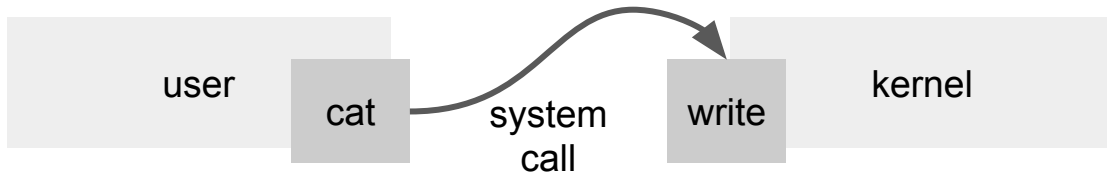


cat.c
if (write(1, buf, n) != n)
defined here

usys.S
SYSCALL(write)

expanded to

```
#define SYSCALL(name) \  
.globl name; \  
name: \  
    movl $SYS_ ## name, %eax; \  
    int $T_SYSCALL; \  
    ret
```



cat.c

```
if (write(1, buf, n) != n)
```

defined here

usys.S

```
SYSCALL(write)
```

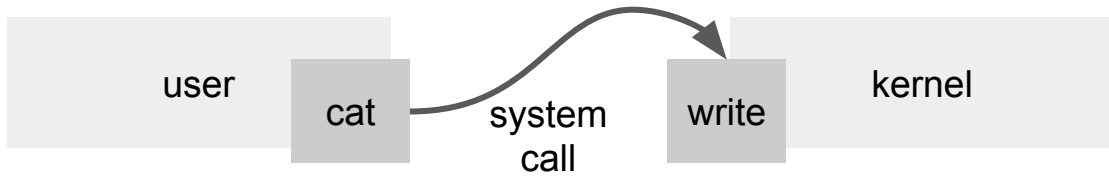
expanded to

```
#define SYSCALL(name) \  
.globl name; \  
name: \  
    movl $SYS_ ## name, %eax; \  
    int $T_SYSCALL; \  
    ret
```

actual assembly
code generated here

usys.o

```
00000028 <write>:  
28:b8 10 00 00 00    mov    $0x10,%eax  
2d:cd 40             int    $0x40  
2f:c3             ret
```



cat.c

```
if (write(1, buf, n) != n)
```

defined here

usys.S

```
SYSCALL(write)
```

expanded to

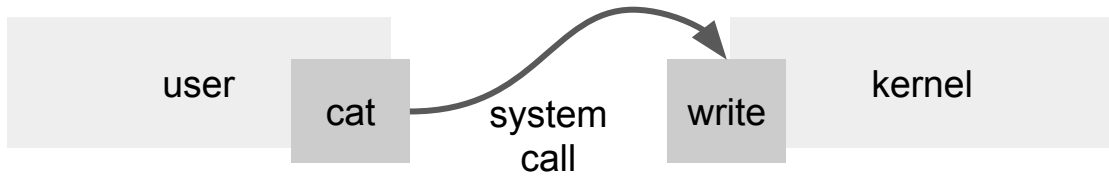
```
#define SYSCALL(name) \  
.globl name; \  
name: \  
    movl $SYS_ ## name, %eax; \  
    int $T_SYSCALL; \  
    ret
```

actual assembly code generated here

usys.o

```
00000028 <write>:  
28:b8 10 00 00 00    mov    $0x10,%eax  
2d:cd 40              int    $0x40  
2f:c3              ret
```

EAX
10



cat.c

```
if (write(1, buf, n) != n)
```

defined here

usys.S

```
SYSCALL(write)
```

expanded to

```
#define SYSCALL(name) \  
.globl name; \  
name: \  
    movl $SYS_ ## name, %eax; \  
    int $T_SYSCALL; \  
    ret
```

actual assembly
code generated here

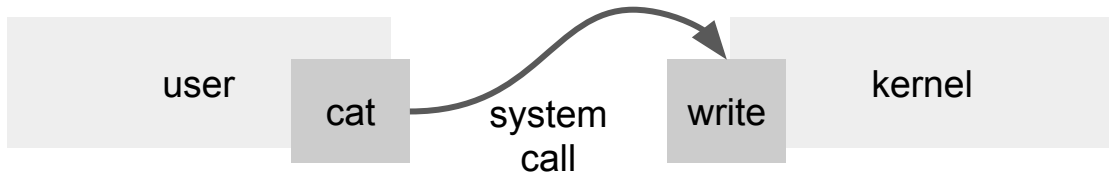
usys.o

```
00000028 <write>:  
28:b8 10 00 00 00    mov    $0x10,%eax  
2d:cd 40              int    $0x40  
2f:c3              ret
```

EAX
10

vectors.S

```
jmp alltraps
```



cat.c

```
if (write(1, buf, n) != n)
```

defined here

usys.S

```
SYSCALL(write)
```

expanded to

```
#define SYSCALL(name) \  
.globl name; \  
name: \  
    movl $SYS_ ## name, %eax; \  
    int $T_SYSCALL; \  
    ret
```

actual assembly code generated here

usys.o

```
00000028 <write>:  
28:b8 10 00 00 00    mov    $0x10,%eax  
2d:cd 40              int    $0x40  
2f:c3              ret
```

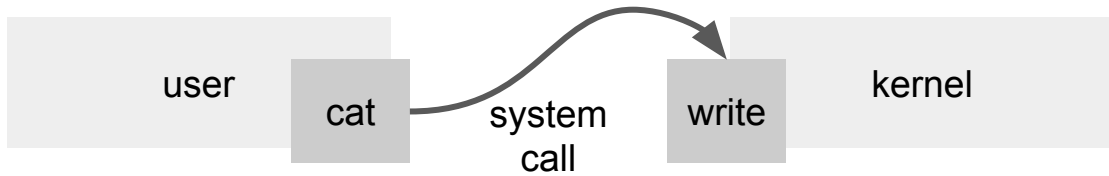
EAX
10

[trapasm.S](#)

defined in

vectors.S

```
jmp alltraps
```



cat.c

```
if (write(1, buf, n) != n)
    defined here
```

usys.S

```
SYSCALL(write)

#define SYSCALL(name) \
    .globl name; \
    name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret
```

expanded to

actual assembly code generated here

usys.o

```
00000028 <write>:
    28:b8 10 00 00 00    mov    $0x10,%eax
    2d:cd 40              int   $0x40
    2f:c3              ret
```

EAX
10

[trap.c](#)

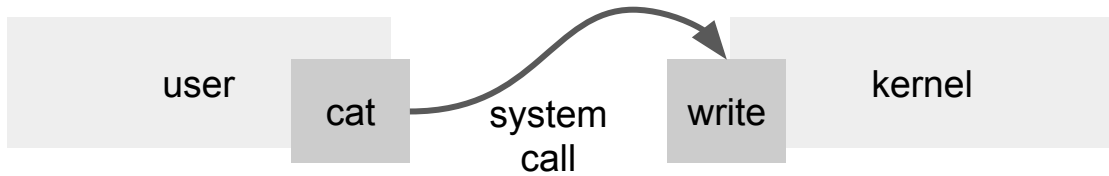
calls trap()

[trapasm.S](#)

defined in

vectors.S

```
jmp alltraps
```



cat.c

```
if (write(1, buf, n) != n)
    defined here
```

usys.S

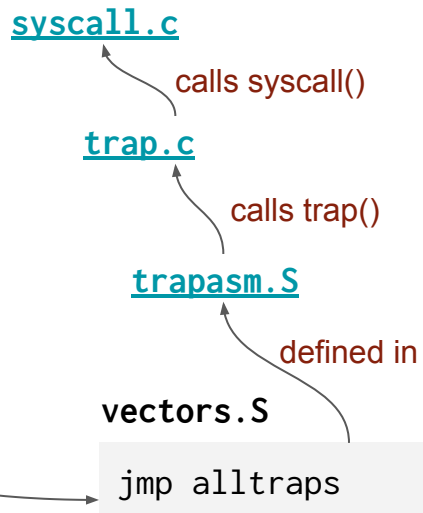
```
SYSCALL(write)
    expanded to
#define SYSCALL(name) \
    .globl name; \
    name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret
```

actual assembly code generated here

usys.o

```
00000028 <write>:
    28:b8 10 00 00 00    mov    $0x10,%eax
    2d:cd 40              int   $0x40
    2f:c3              ret
```

EAX
10



Thank you!