Process Messages

Kernel

a computer program

Kernel

a computer program

that is **the core** of a computer's operating system

Kernel

a computer program

that is **the core** of a computer's operating system

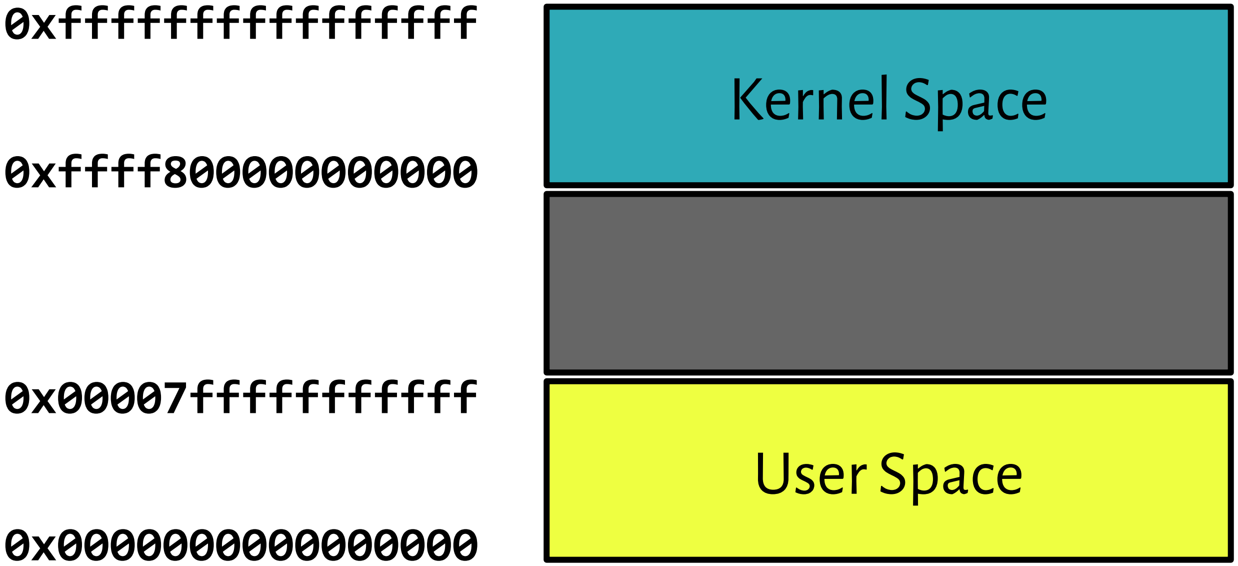
with **complete control over everything** in the system

Kernel

a computer program

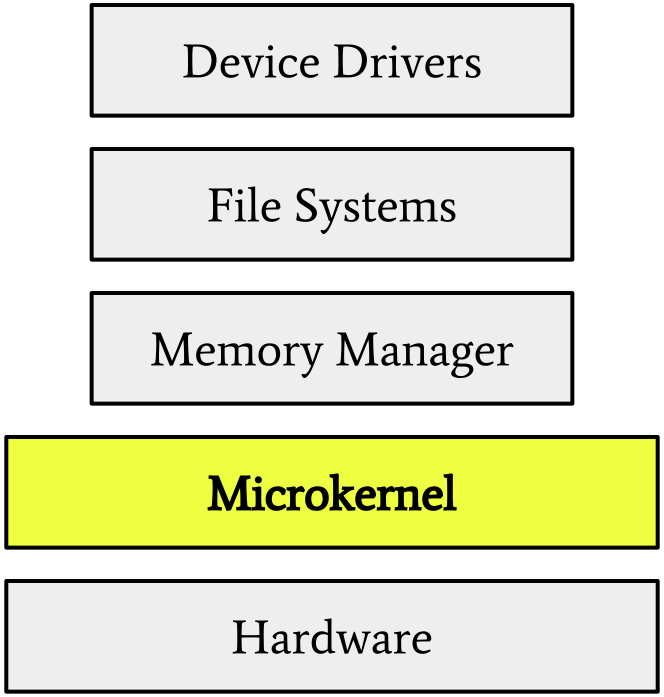
that is **the core** of a computer's operating system

with **complete control over everything** in the system



Microkernel vs monolithic system

Microkernel vs monolithic system



Device Drivers

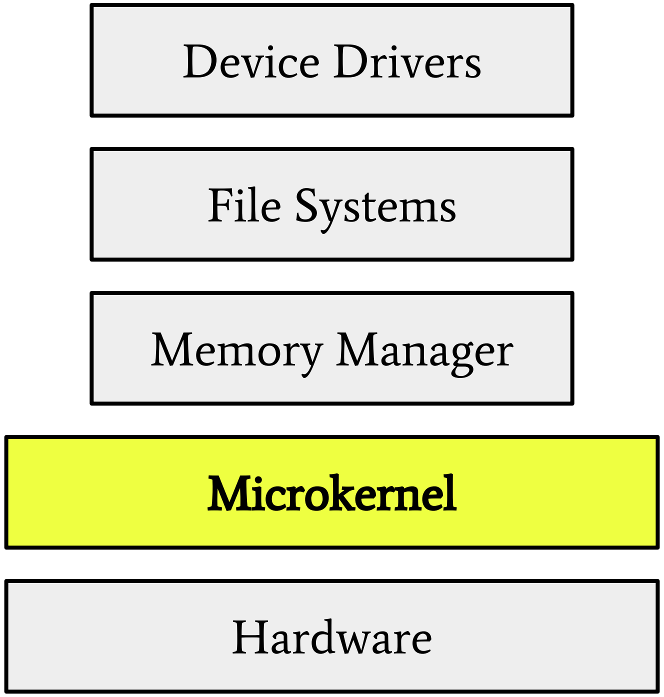
File Systems

Memory Manager

Microkernel

Hardware

Microkernel vs monolithic system



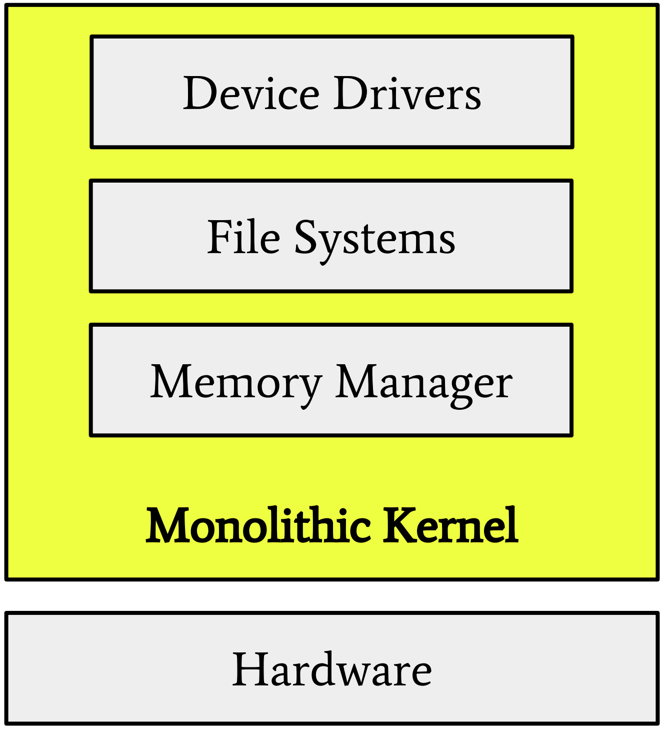
Device Drivers

File Systems

Memory Manager

Microkernel

Hardware



Device Drivers

File Systems

Memory Manager

Monolithic Kernel

Hardware

Microkernel vs monolithic system

“To me, writing a monolithic system in 1991 is a truly poor idea.”

Microkernel vs monolithic system

“To me, writing a monolithic system in 1991 is a truly poor idea.”

*“I've got more excuses than you have, and **Linux still beats the pants of minix in almost all areas**.”*

Microkernel vs monolithic system

“To me, writing a monolithic system in 1991 is a truly poor idea.”

*“I've got more excuses than you have, and **Linux still beats the pants of minix in almost all areas**.”*

*“That's one hell of a good excuse for some of the brain-damages of minix. I can only hope (and assume) that **Amoeba doesn't suck like minix does**.”*

Microkernel vs monolithic system

“To me, writing a monolithic system in 1991 is a truly poor idea.”

*“I've got more excuses than you have, and **Linux still beats the pants of minix in almost all areas**.”*

*“That's one hell of a good excuse for some of the brain-damages of minix. I can only hope (and assume) that **Amoeba doesn't suck like minix does**.”*

Christoph Lameter, *Extreme High Performance Computing or Why Microkernels Suck*

https://www.kernel.org/doc/ols/2007/ols2007v1-pages-251-262.pdf

Why is microkernel ineffective?

Task G1

Task G1

```
# ps axl | grep rs
```

Task G1

```
# ps axl | grep rs
```

```
# ps axl | grep 4
```

Task G1

```
# ps axl | grep rs
```

```
# ps axl | grep 4 | tee /tmp/info
```

Task G1

```
# ps axl | grep rs
```

```
# ps axl | grep 4 | tee /tmp/info
```

```
# vim /tmp/info
```

Why is microkernel ineffective?

Why is microkernel ineffective?

```
# cat /usr/src/minix/lib/libc/sys/syscall.c
```

```c
int _syscall(endpoint_t who, int syscallnr, message *msgptr)
{
    int status;
    msgptr->m_type = syscallnr;
    status = ipc_sendrec(who, msgptr);

    if (status != 0) {
        /* 'ipc_sendrec' itself failed. */
        /* XXX - strerror doesn't know all the codes */
        msgptr->m_type = status;
    }

    if (msgptr->m_type < 0) {
        errno = -msgptr->m_type;
        return(-1);
    }

    return(msgptr->m_type);
}
```

Task G2

Task G2

```
# vim /usr/src/minix/lib/libc/gen/wait.c
```

Task G2

```
# vim /usr/src/minix/lib/libc/gen/wait.c
```

```
# grep -nri PM_WAITPID /usr/src/minix

# vim /usr/src/minix/servers/pm/table.c
```

Task G2

```
# vim /usr/src/minix/lib/libc/gen/wait.c
```

```
# grep -nri PM_WAITPID /usr/src/minix

# vim /usr/src/minix/servers/pm/table.c
```

```
# grep -nri do_waitpid /usr/src/minix

# vim /usr/src/minix/servers/pm/forkexit.c
```

Task G3

Task G3

```
# vim /usr/src/minix/servers/ipc/inc.h
```

Task G3

```
# vim /usr/src/minix/servers/ipc/inc.h
```

```
# grep -nri do_semop /usr/src/minix

# vim /usr/src/minix/servers/ipc/sem.c
```

Task G3

```
# vim /usr/src/minix/servers/ipc/inc.h
```

```
# grep -nri do_semop /usr/src/minix

# vim /usr/src/minix/servers/ipc/sem.c
```

The general idea:

In order to suspend a process we let it wait for a response for its syscall message.

Assignment #3

/home/students/inf/PUBLIC/SO/zadania/3/zad3.html