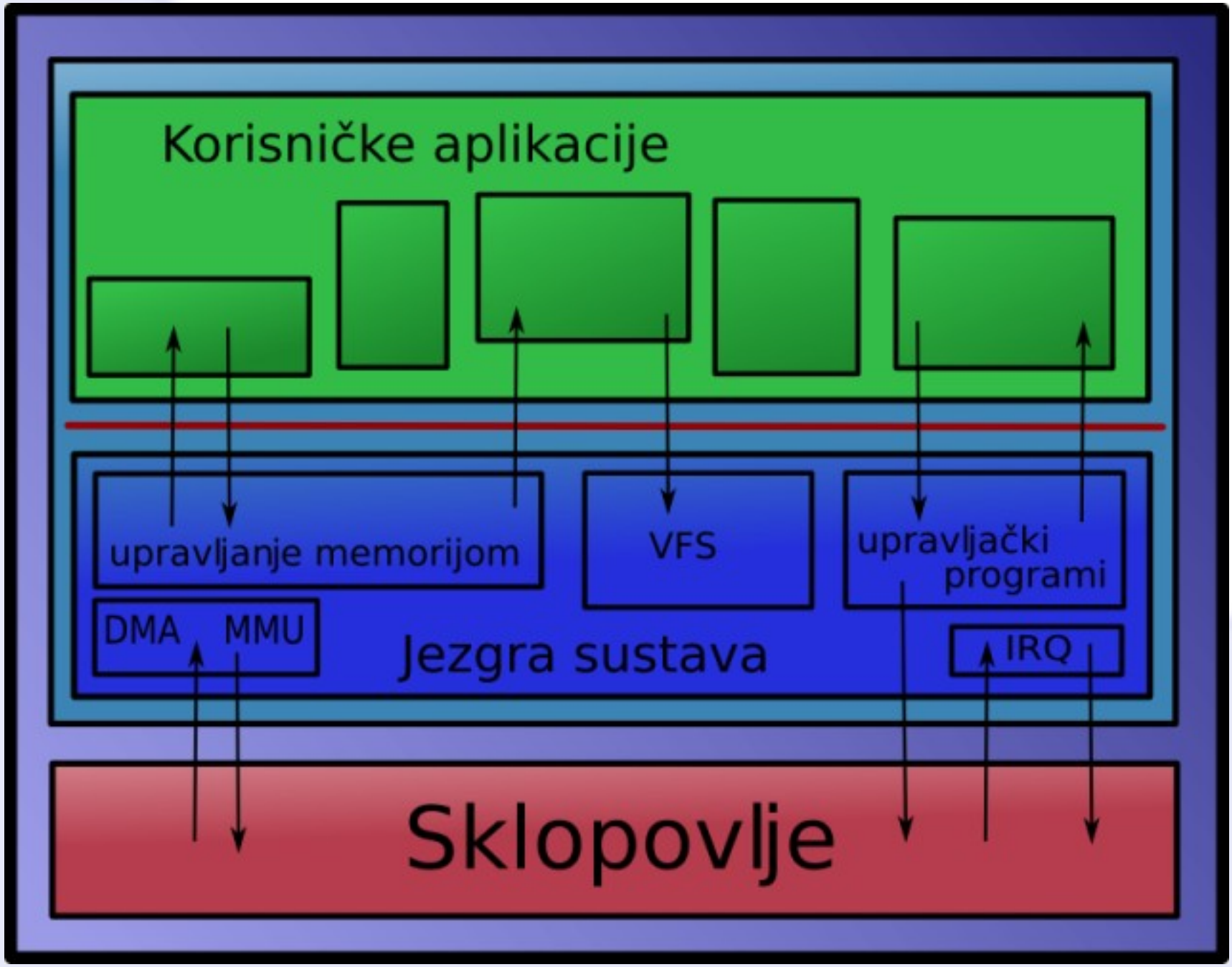
Microkernel-based operating system development

Senko Rašić

operating system (OS):

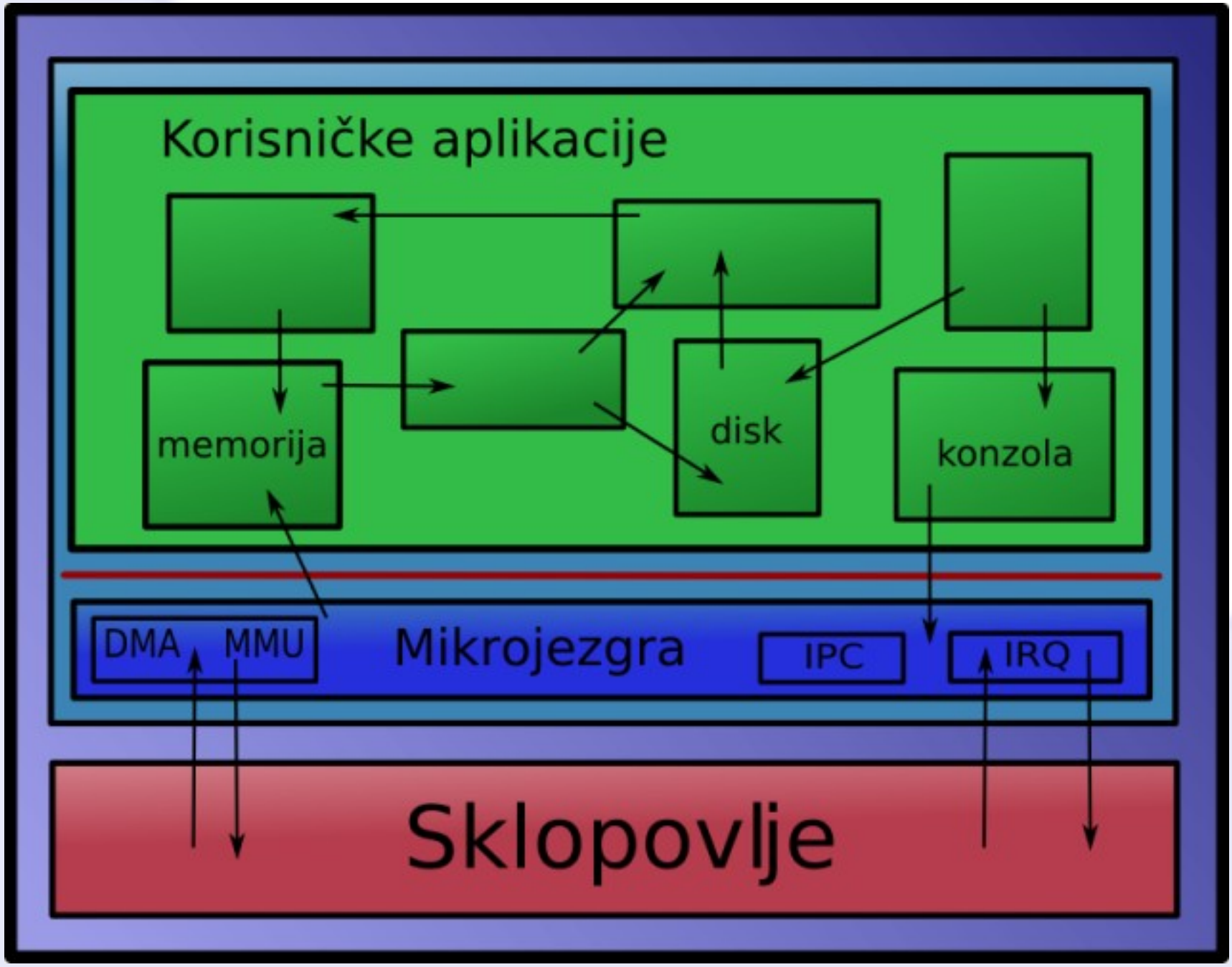
A part of software that manages hardware resources and provides a safe environment for processes to execute in.

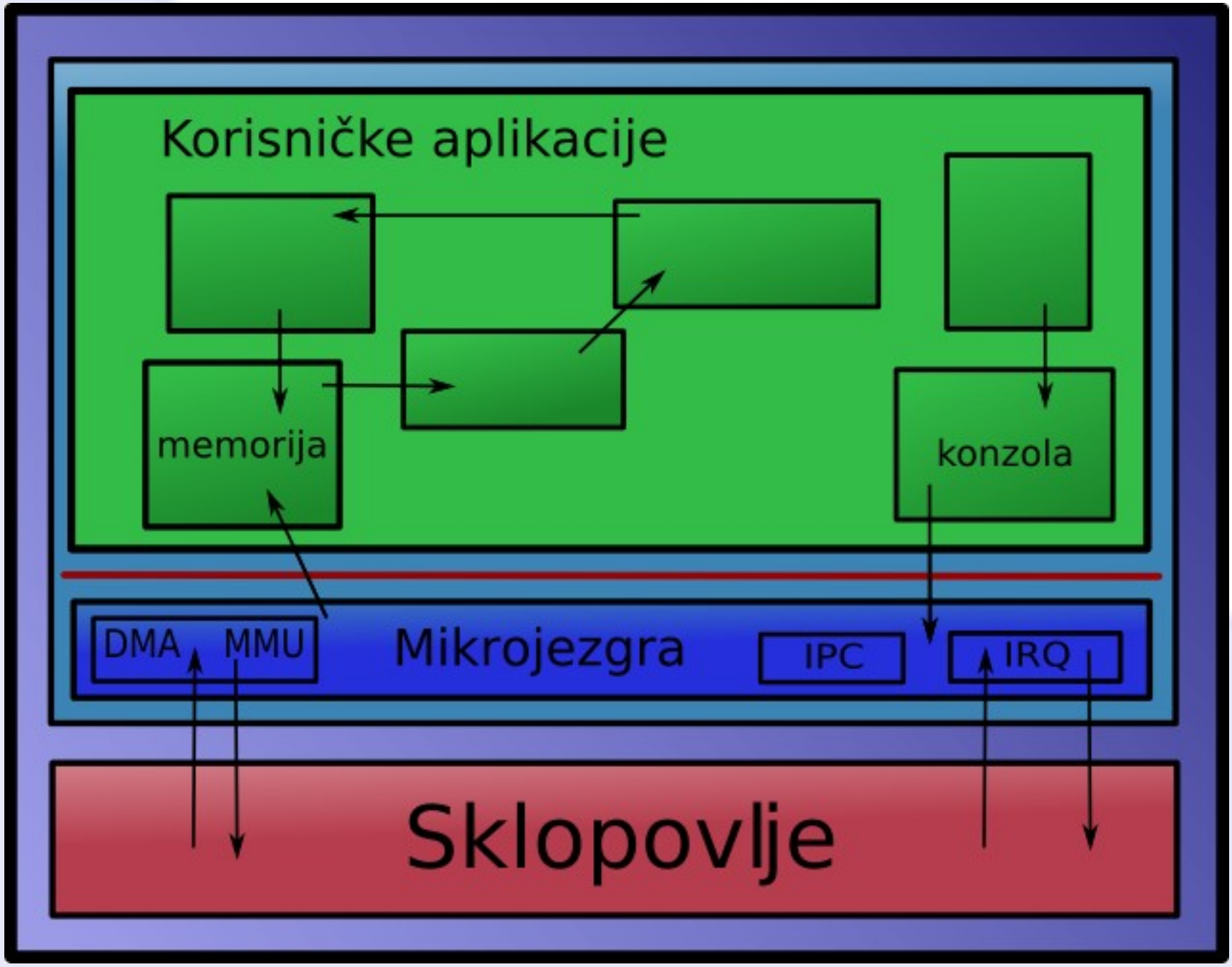

Korisničke aplikacije
upravljanje memorijom
VFS
upravljački programi
DMA
MMU
Jezgra sustava
IRQ
Sklopovlje

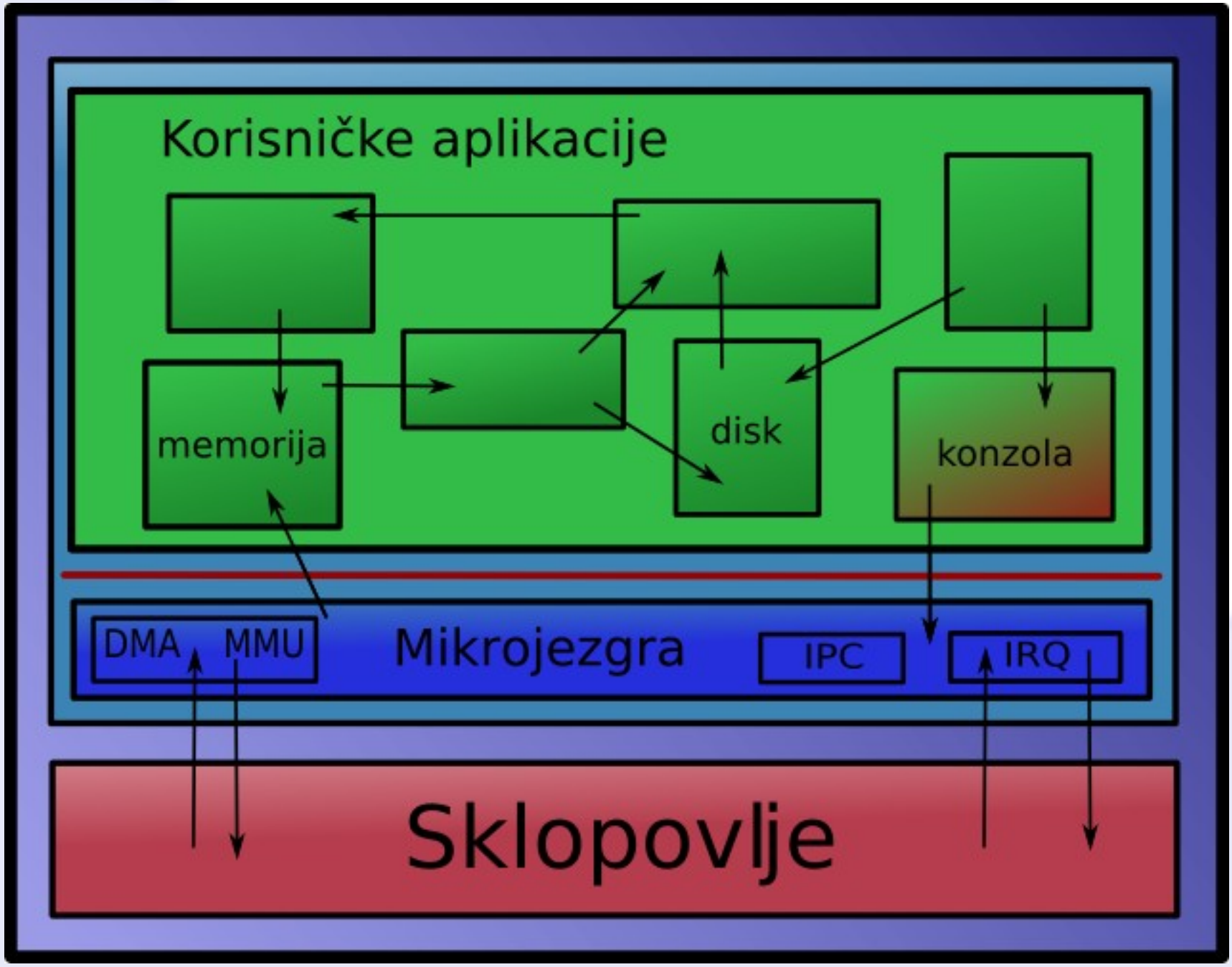
the micro-kernel idea:

Hardware drivers and operating system services are **regular processes**.

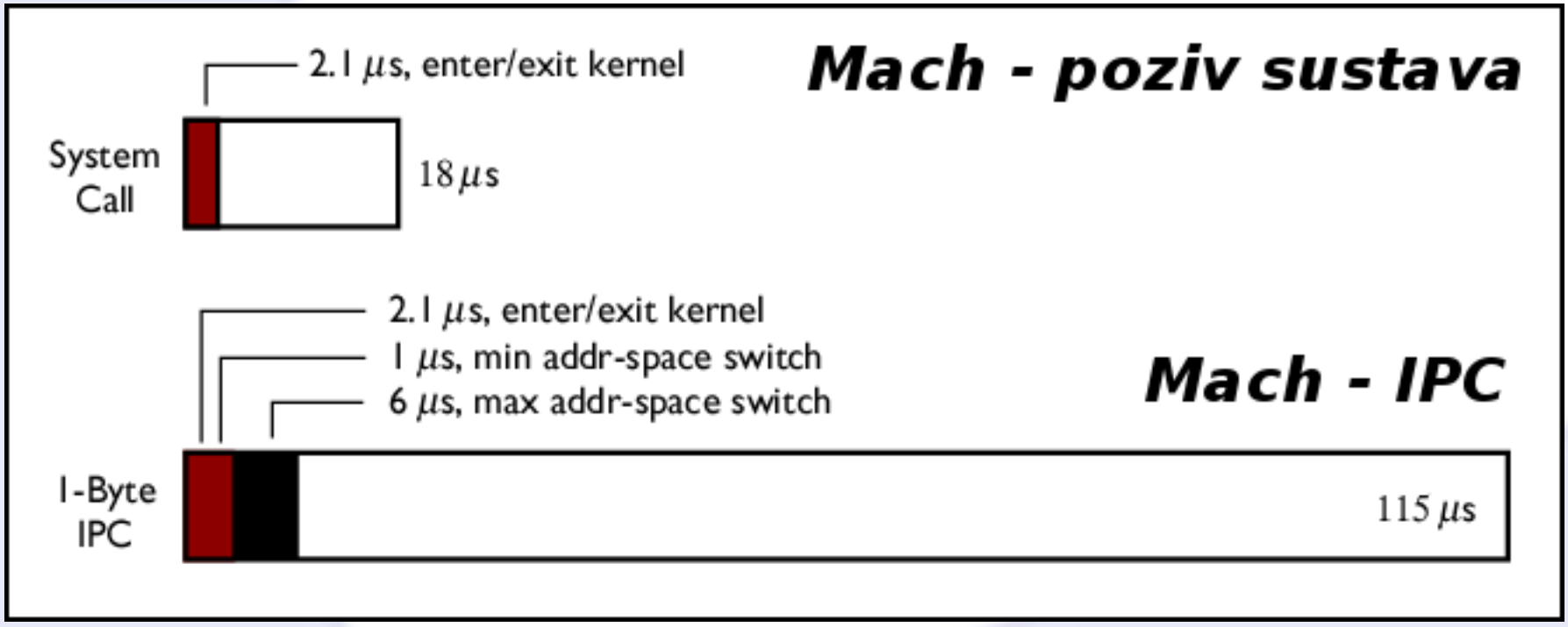
The kernel has only the required basic functionality


Korisničke aplikacije
memorija
disk
konzola
DMA
MMU
Mikrojezgra
IPC
IRQ
Sklopovlje


Korisničke aplikacije
memorija
konzola
DMA
MMU
Mikrojezgra
IPC
IRQ
Sklopovlje


Korisničke aplikacije
memorija
disk
konzola
DMA
MMU
Mikrojezgra
IPC
IRQ
Sklopovlje

intensive inter-process comunication + slow IPC calls



= bad performance

mainstream systems:

- Monolithic kernels with some elements of microkernel design

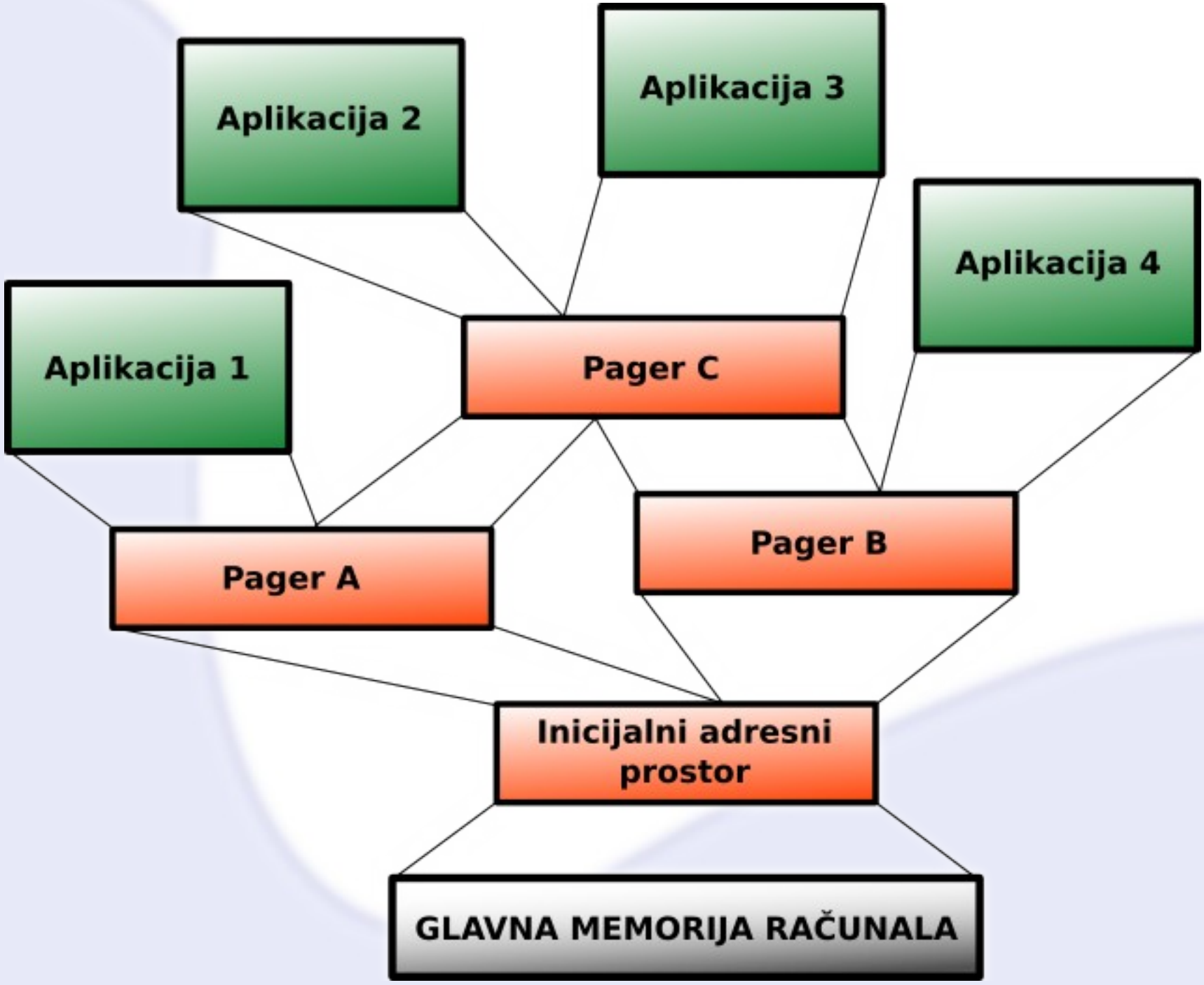
ongoing research:

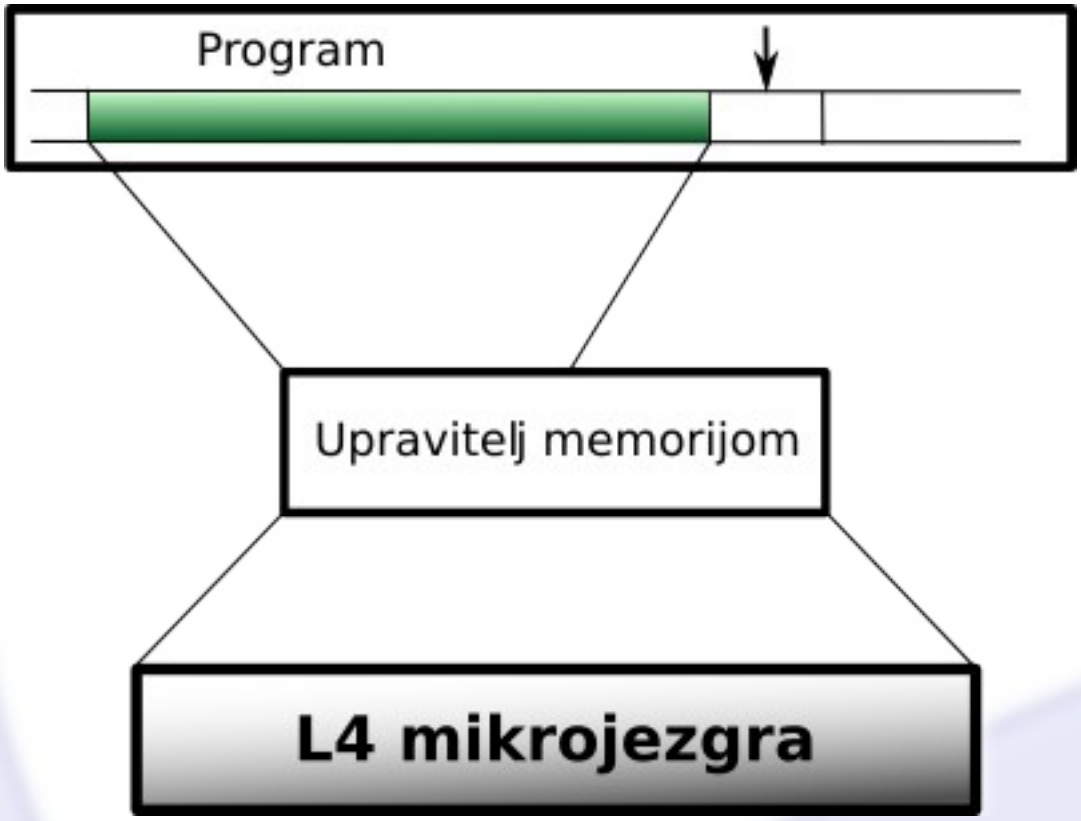
- L4, Exokernel, ...

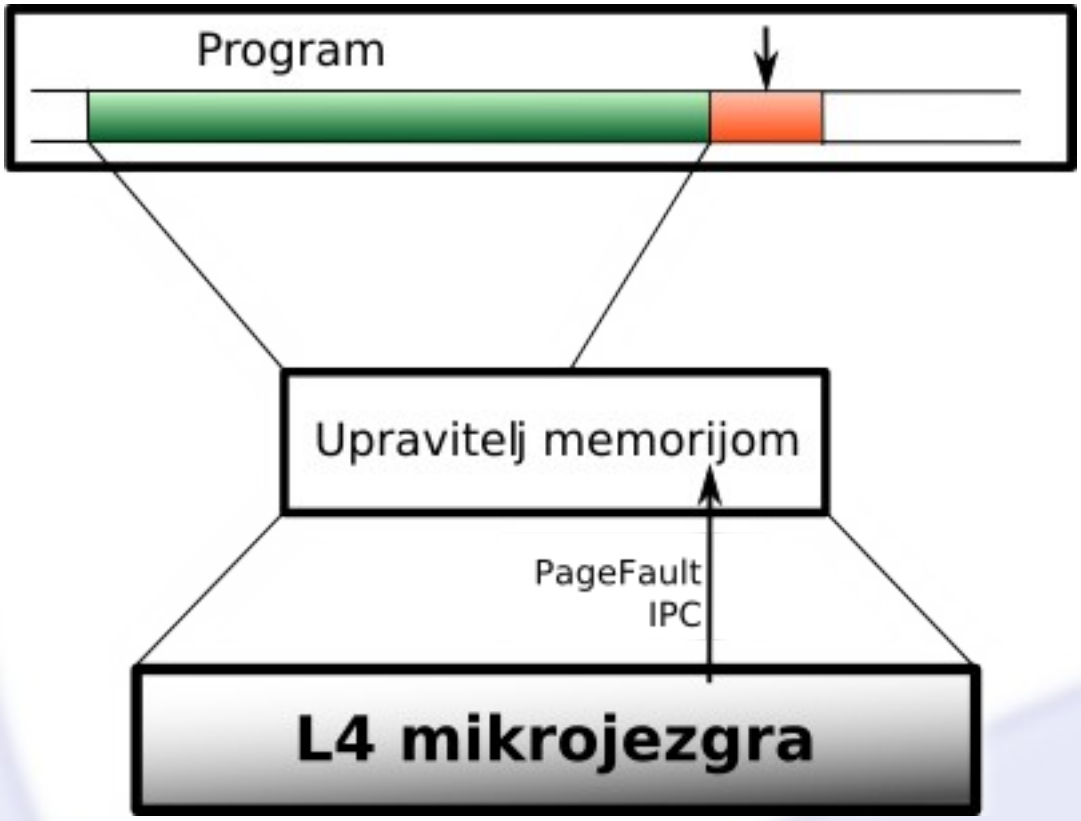
L4 microkernel

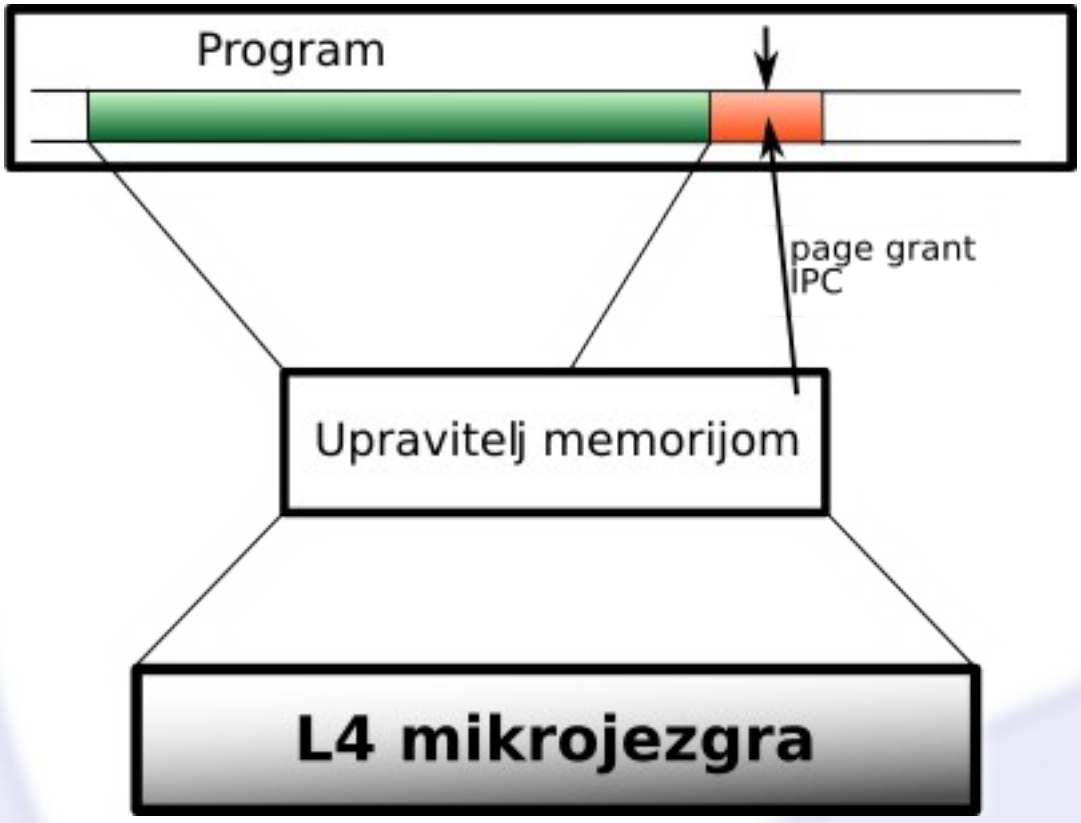
- recursive address spaces
- highly optimized IPC

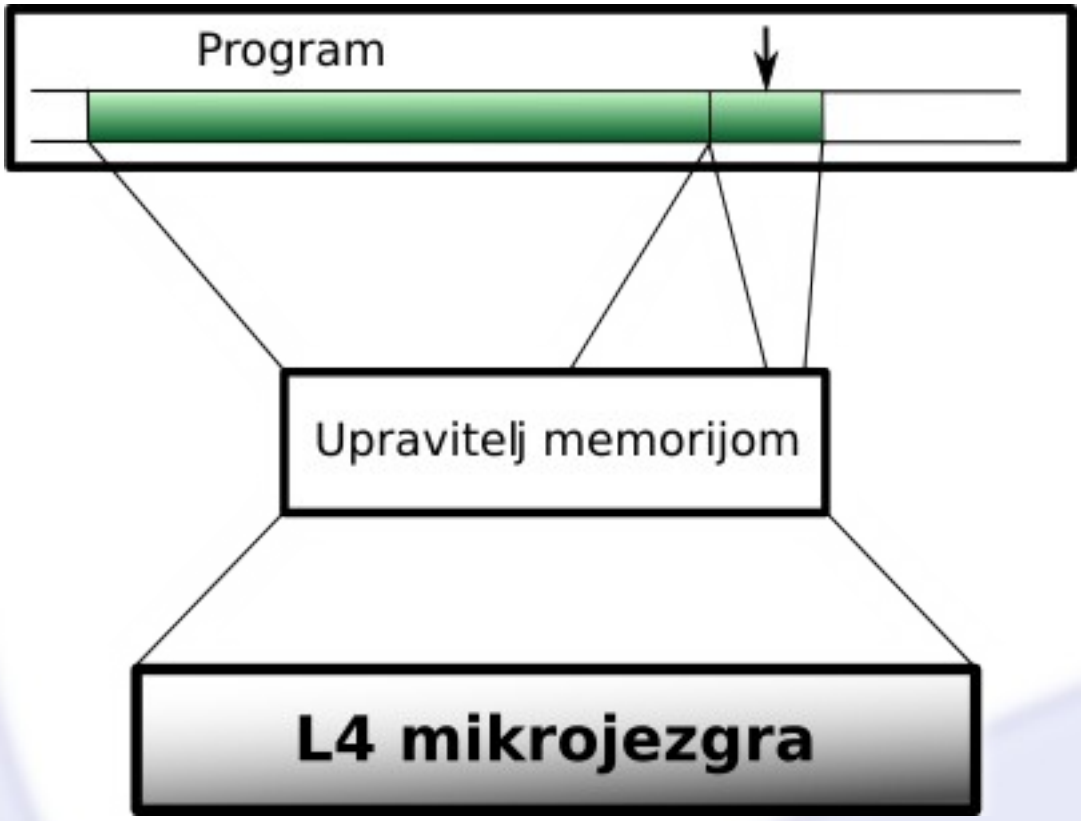
L4Ka::Pistachio – an implementation by the University of Karlsruhe


Aplikacija 2
Aplikacija 3
Aplikacija 4
Aplikacija 1
Pager C
Pager A
Pager B
Inicijalni adresni prostor
GLAVNA MEMORIJA RAČUNALA


Program
Upravitelj memorijom
L4 mikrojezgra


Program
Upravitelj memorijom
PageFault
IPC
L4 mikrojezgra


Program
page grant
IPC
Upravitelj memorijom
L4 mikrojezgra

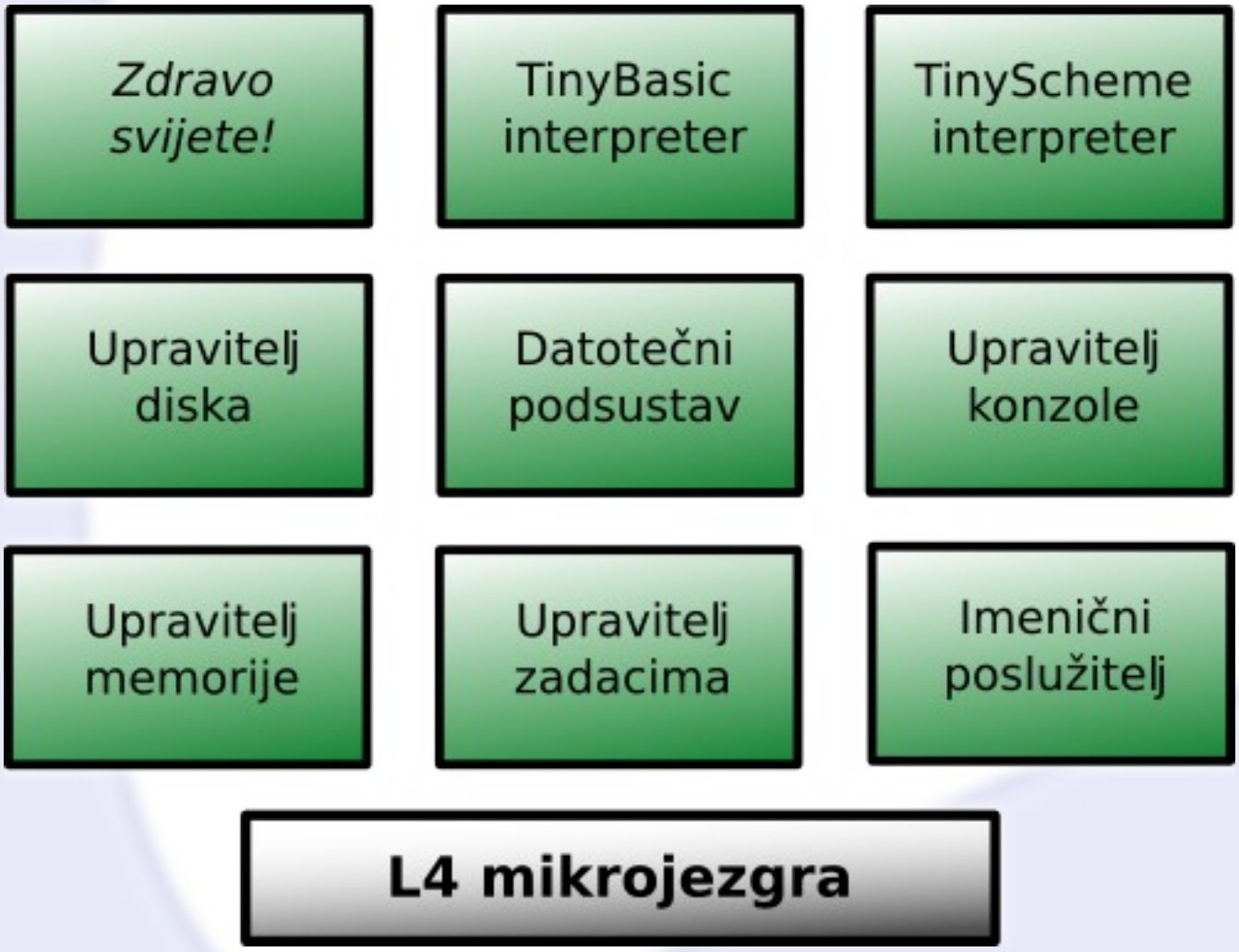

Program
Upravitelj memorijom
L4 mikrojezgra

Open Source software

- availability
- community
- malleability

Hasenpfeffer

- based on L4 microkernel
- set of servers cooperating through IPC calls
- maximal reuse of available open source components (NIH syndrome avoidance)

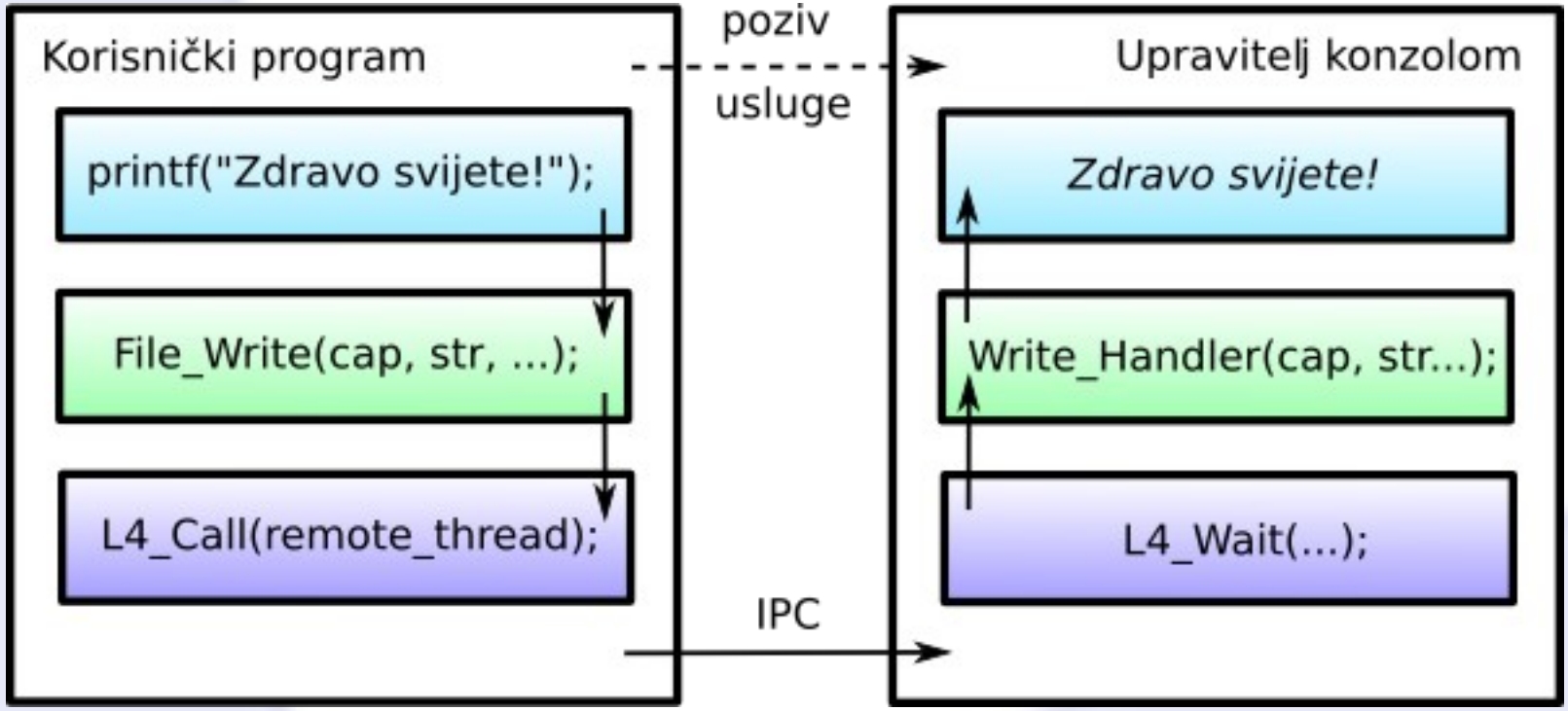

Zdravo svijete!
TinyBasic interpreter
TinyScheme interpreter
Upravitelj diska
Datotečni podsustav
Upravitelj konzole
Upravitelj memorije
Upravitelj zadacima
Imenični poslužitelj
L4 mikrojezgra

Inter-process communication

- server and client processes
- communication interface
- IDL[4] generates server and client stubs

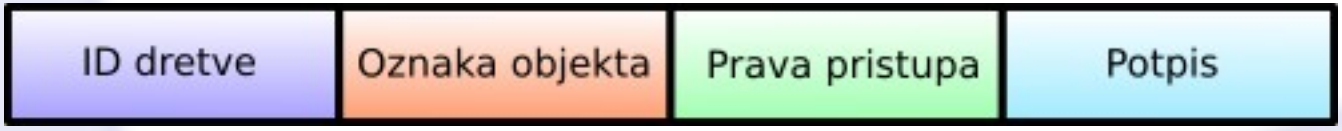
```
interface File {

   boolean Read(in Capability_t cap,in unsigned long
     offset, out byteseq_t buffer, inout unsigned long
     size);

   unsigned long Size(in Capability_t cap);

   ...

}
```

Remote procedure calls

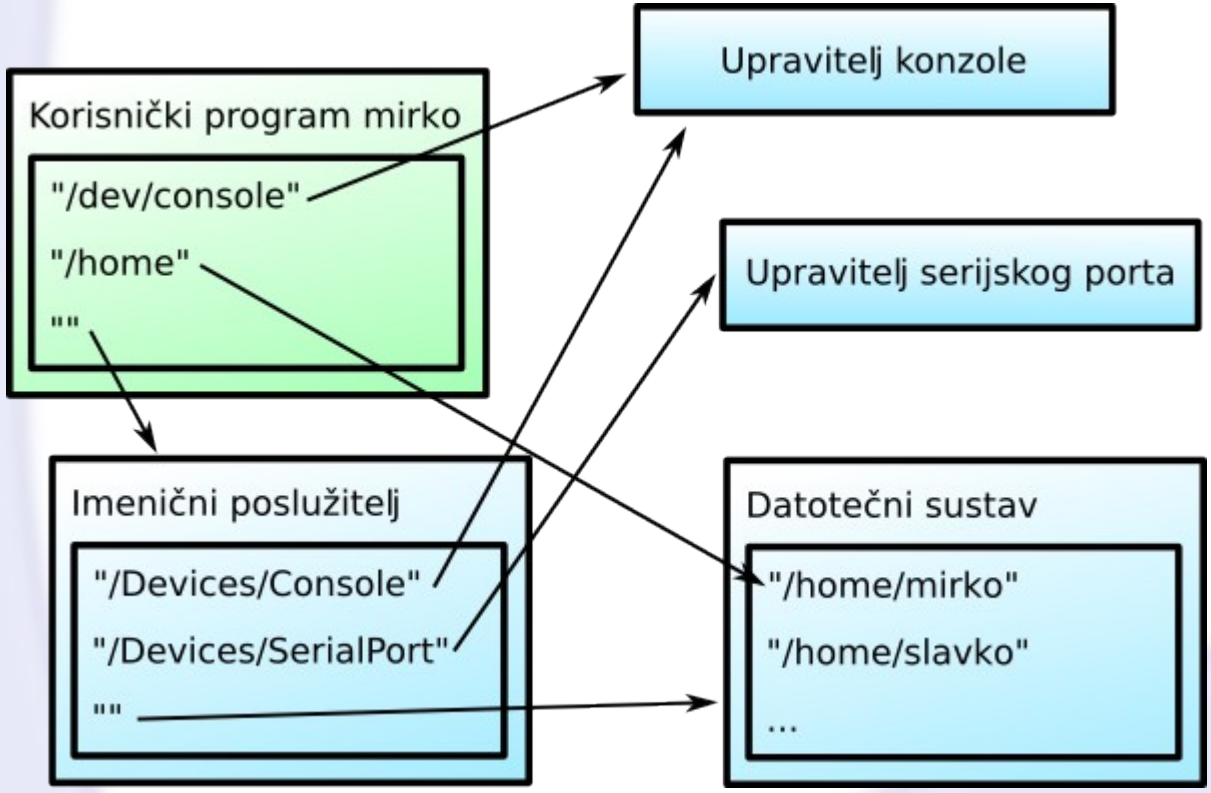


capability:

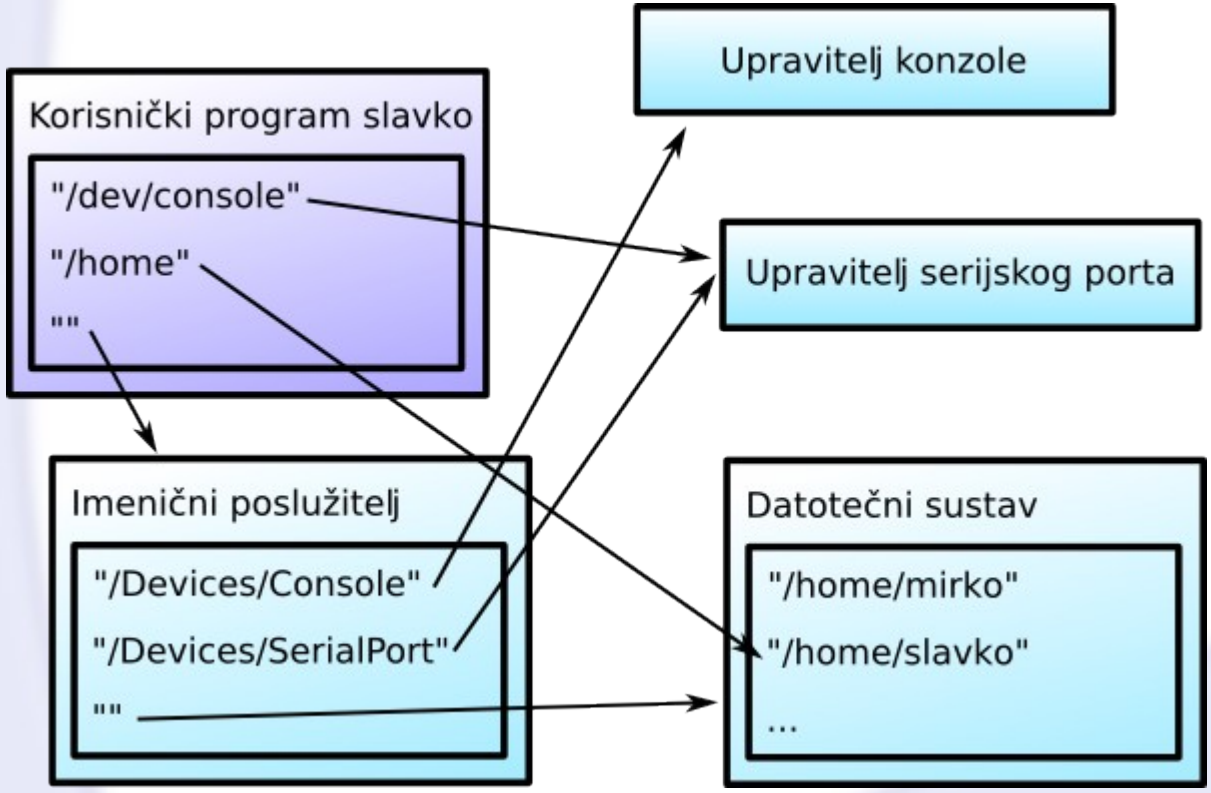
The privilege to invoke a specific RPC.



Naming system



Name system


Upravitelj konzole
Korisnički program slavko
"/dev/console"
"/home"
""
Upravitelj serijskog porta
Imenični poslužitelj
"/Devices/Console"
"/Devices/SerialPort"
""
Datotečni sustav
"/home/mirko"
"/home/slavko"
...

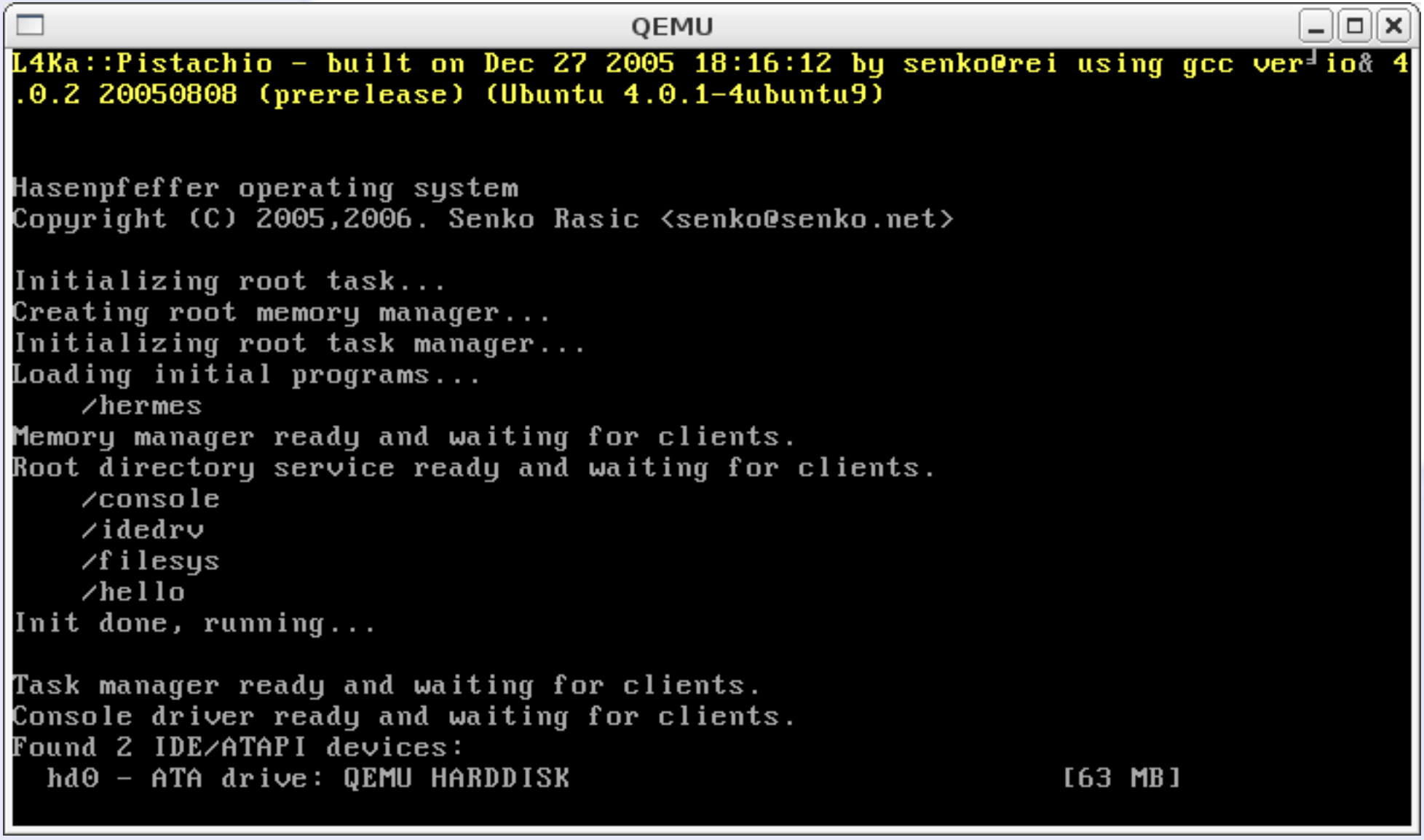
Features

- program loading and execution (*ELF*)
- multiprocessing, multithreading
- standard C library available
- hard disk (*IDE*) support
- TinyScheme i TinyBasic programming languages

3rd party components

- IDE disk driver
- display output, keyboard input
- C library
- TinyScheme interpreter
- TinyBasic interpreter

System boot


QEMU
L4Ka::Pistachio - built on Dec 27 2005 18:16:12 by senko@rei using gcc ver io& 4
.0.2 20050808 (prerelease) (Ubuntu 4.0.1-4ubuntu9)
Hasenpfeffer operating system
Copyright (C) 2005,2006. Senko Rasic <senko@senko.net>
Initializing root task...
Creating root memory manager...
Initializing root task manager...
Loading initial programs...
/hermes
Memory manager ready and waiting for clients.
Root directory service ready and waiting for clients.
/console
/idedrv
/filesys
/hello
Init done, running...
Task manager ready and waiting for clients.
Console driver ready and waiting for clients.
Found 2 IDE/ATAPI devices:
hd0 - ATA drive: QEMU HARDDISK [63 MB]

Software Development Kit

- Linux system
- GNU C/C++ compiler
- L4Ka::Pistachio development environment

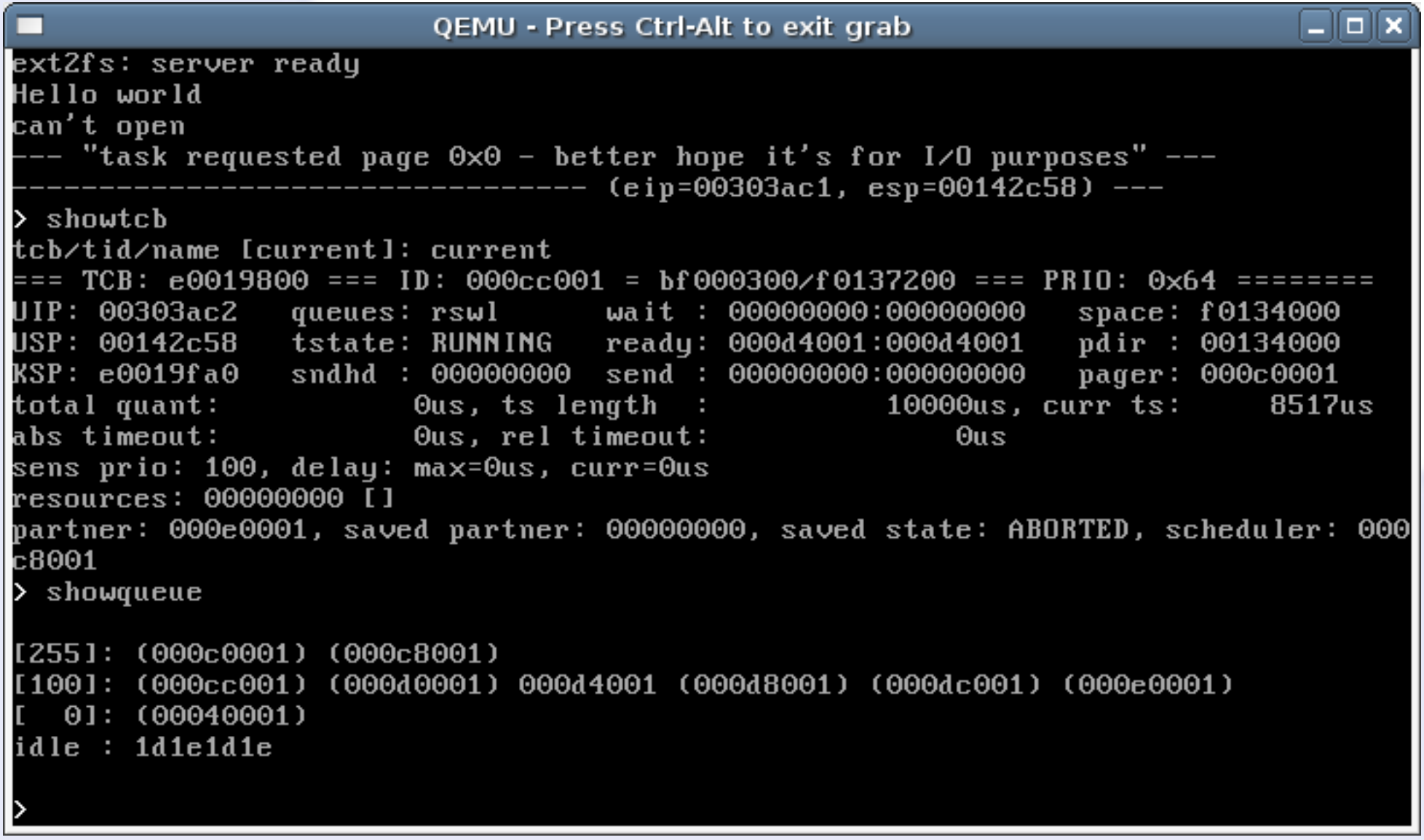
Testing

- QEMU
- VMWare Workstation
- L4 KDB (*Kernel Debugger*)

Test environment:

- floppy disk image for OS boot (using grub)
- IDE disk image with ext2 filesystem

QEMU & KDB in action



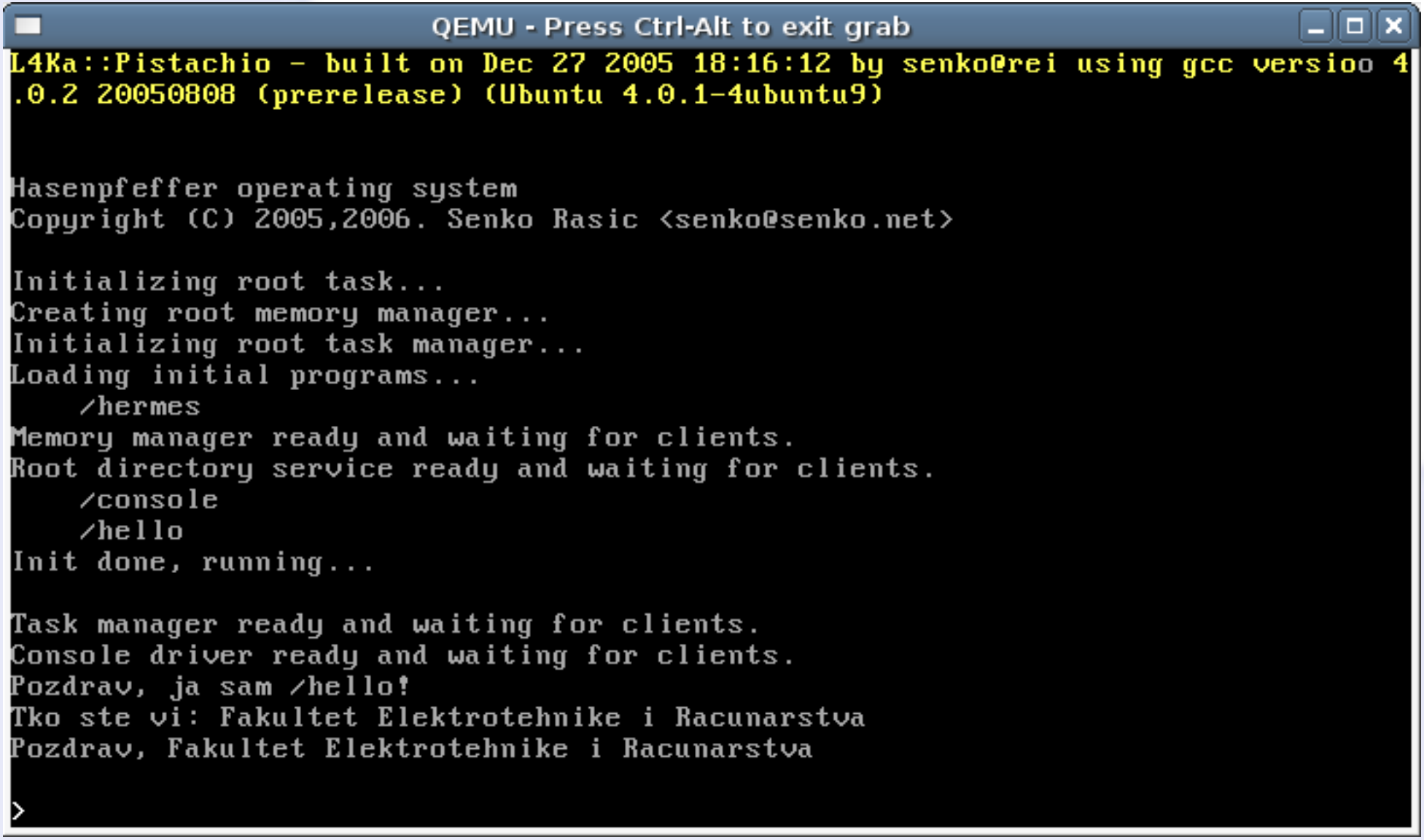
Software development

```
#include <stdio.h>
#include <l4/ipc.h>

#define SECOND 1000000UL

int main(int argc, char *argv[])
{
    char buf[1024];

    L4_Sleep(L4_TimePeriod(4 * SECOND));
    printf("Hello, I'm %s!\nWho are you: ", argv[0]);

    fgets(buf, 1023, stdin);
    printf("Howdy, %s\n", buf);

    return 0;
}
```

Software development

- Standard C source code
- Build system config (scons)
  - based on L4 system build system
- Build and creation of floppy image
- System boot configuration
- Virtual machine execution and testing

Software development



In summary

developed system:

- rudimentary
- extensable

microkernel:

- good base for future development

open source:

- availability
- malleability