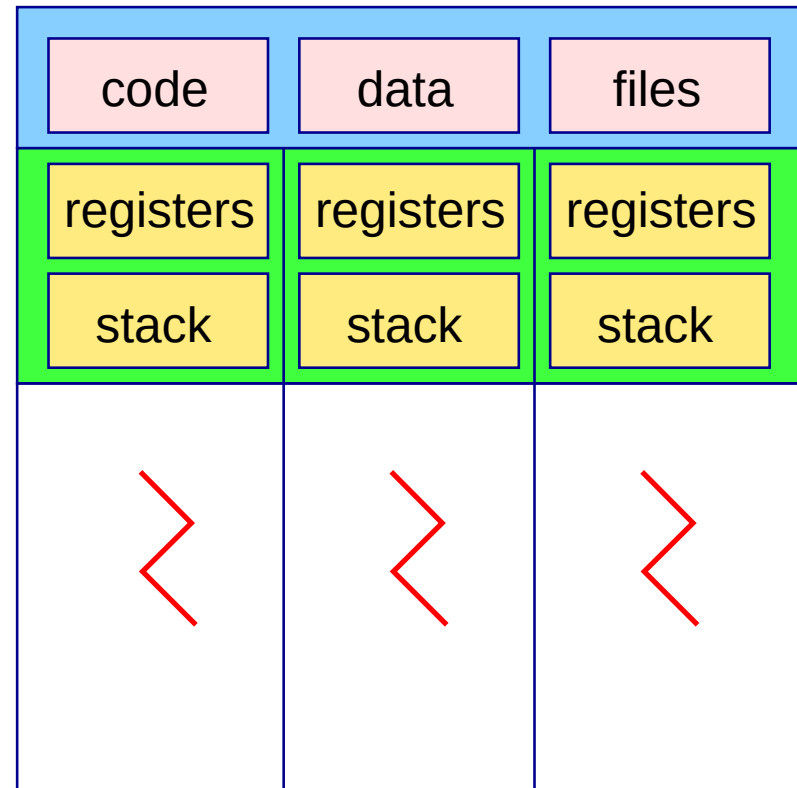
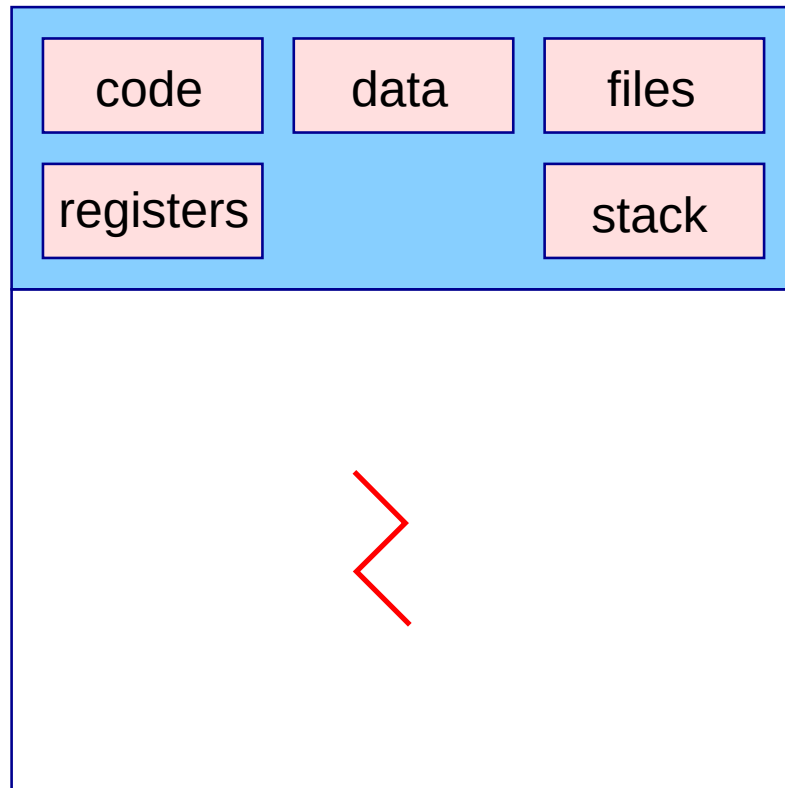


Lõimed

# Ülevaade

- Lõime mõiste
- Lõimede mudelid
- Probleemid lõimedega seoses
- Pthreads
- Solarise lõimed
- Windows 2000 lõimed
- FreeBSD lõimed
- Linux'i lõimed

# Ühe- ja mitmelõimelised protsessid



## Lõimed: kolm vaatepunkti

- Abstraktsioon algoritmide esitamiseks
- Programmeerimisliides
  - Alternatiiv: olekuautomaadid
- Operatsioonisüsteemi ajajaotusühik
  - Alternatiiv: lihtsalt protsessid

## Lõimed — milleks?

- Reageerimiskiirus — parema interaktiivsuse saamiseks võib kasutajaliidese ja muu töö eraldi lõimedesse panna
- Ressursside jagamine — aadressipiirkond programmi mitme täitja poolt kasutatav
- Kokkuvõtteid — hoiame kokku jagatud ressursside, uute protsesside loomise ning kontekstivahetuste pealt
- Mitme protsessori ära kasutamine ühe programmi poolt

## Kasutajaseme lõimed

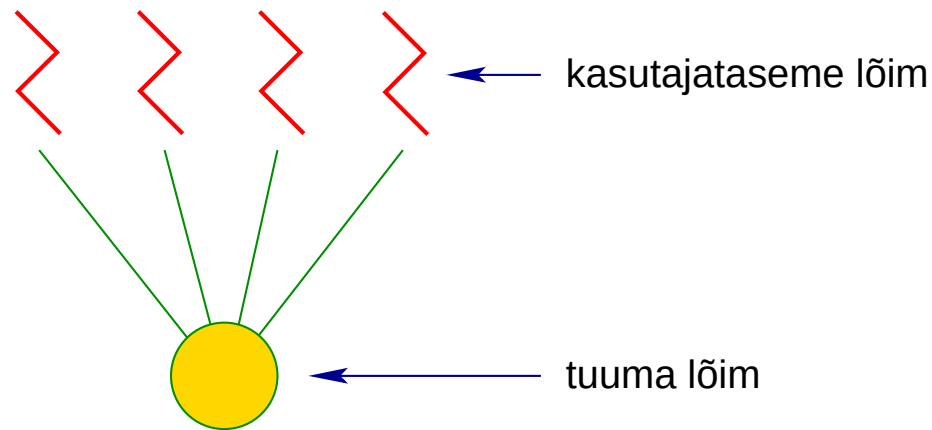
- Lõimede haldus realiseeritud kasutaja tasemel teegina
- Tuum ei tea midagi, tema jaoks üks ühelõimeline protsess
- Haldus kiire, kuna toimub kasutaja tasemel
- Näiteks:
  - POSIXi Pthreads
  - Mach'i C-threads
  - Solarise lõimed (UI-threads)
  - FreeBSD 4 lõimed
  - Windows NT fiibrid (kiud — osa lõimest)

## Tuuma lõimed

- Toetatud tuuma poolt, tuum jagab nende lõimede vahel aega
- Haldus kulukam kui kasutaja tasemel
- Näiteks:
  - Windows 95/98/NT/2000
  - Solaris
  - Tru64 UNIX
  - BeOS
  - Linux

## Lõimede mudelid

- Mitu ühele — protsessi kõigi kasutaja tasemel lõimede jaoks on üks tuuma lõim

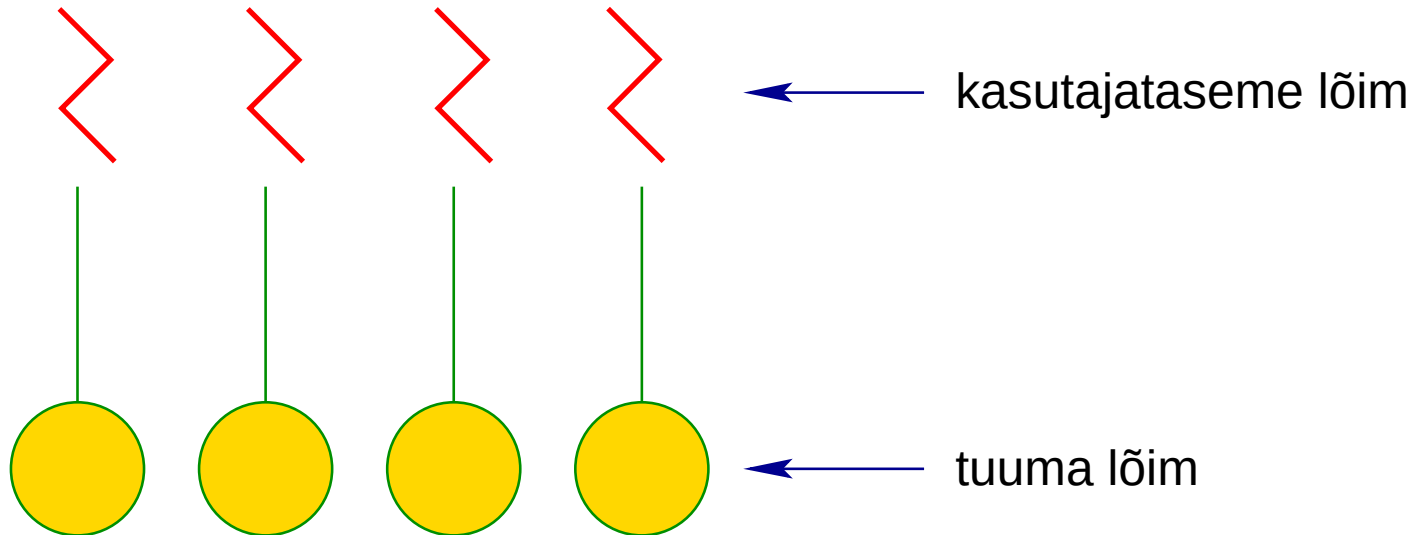


- Haldus kiire, kuna toimub kasutaja tasemel
- Korraga saab töötada kuni üks lõim
- Blokeeruva süsteemifunktsiooni kasutamisel blokeerub kogu protsess



## Lõime mudelid

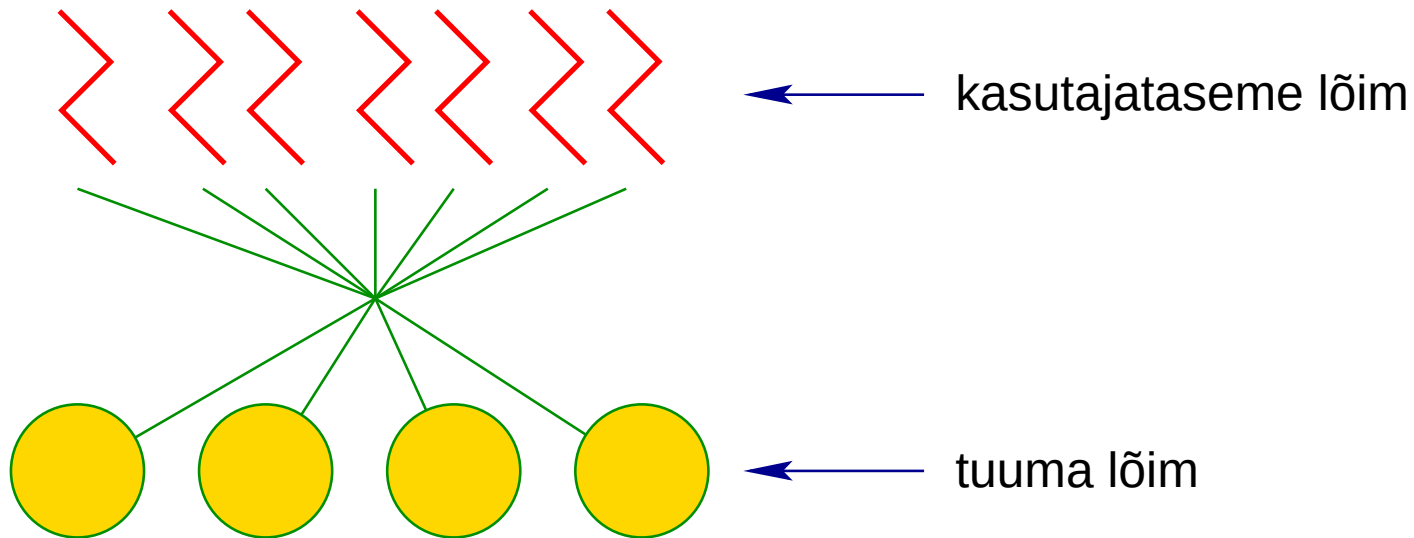
- Üks ühele — igale kasutaja taseme lõimele vastab tuuma lõim



- Haldus kulukam
- Saab kasutada mitut protsessorit
- Saab korraga teha mitut blokeeruvat operatsiooni
- Ei maksa liiga palju lõimi teha

## Lõime mudelid

- Mitu mitmele — protsessi paljudele kasutaja taseme lõimedele vastab palju (kuni samapalju, võib ka vähem) tuuma lõimesid



- Kombineerib eelmiste head küljed
- Keerukam

## Probleemid lõimedega seoses

- `fork()` semantika muutus — kas alamprotsess tehakse paljulõimeline või kopeeritakse üks lõim?
- `exec()` semantika muutus — kas asendatakse kogu protsess või üks lõim?
- Signaalide püüdmine — missugune lõim saab mitmelõimelise protsessi signaalid?
- Lõimede tekitamine ja hävitamine  $\Rightarrow$  lõimevaru (*thread pool*)
- Lõimede lõpetamine (*cancellation*): asünkroonne ja sünkroonne
- Lõime kohalikud andmed (TLS — *thread-local storage*)

## Pthreads

- POSIXi standard (IEEE 1003.1c, POSIX 1c)
- Defineerib programmeerimisliidese lõimedele (C tasemel)
- Kasutaja tasemel, tuuma lõimedega soetud ainult konkreetse implementatsiooni detailide kaudu
- pthread.h, pthread\_\* funktsioonid
- *De facto* standard UNIXites
- Signaalitöötlus kohmakas, muidu normaalne API

## Pthreads näide

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS      5

void *thread_func(void *threadid)
{
    printf("\n%d: Hello World!\n", threadid);
    pthread_exit(NULL);
}
```

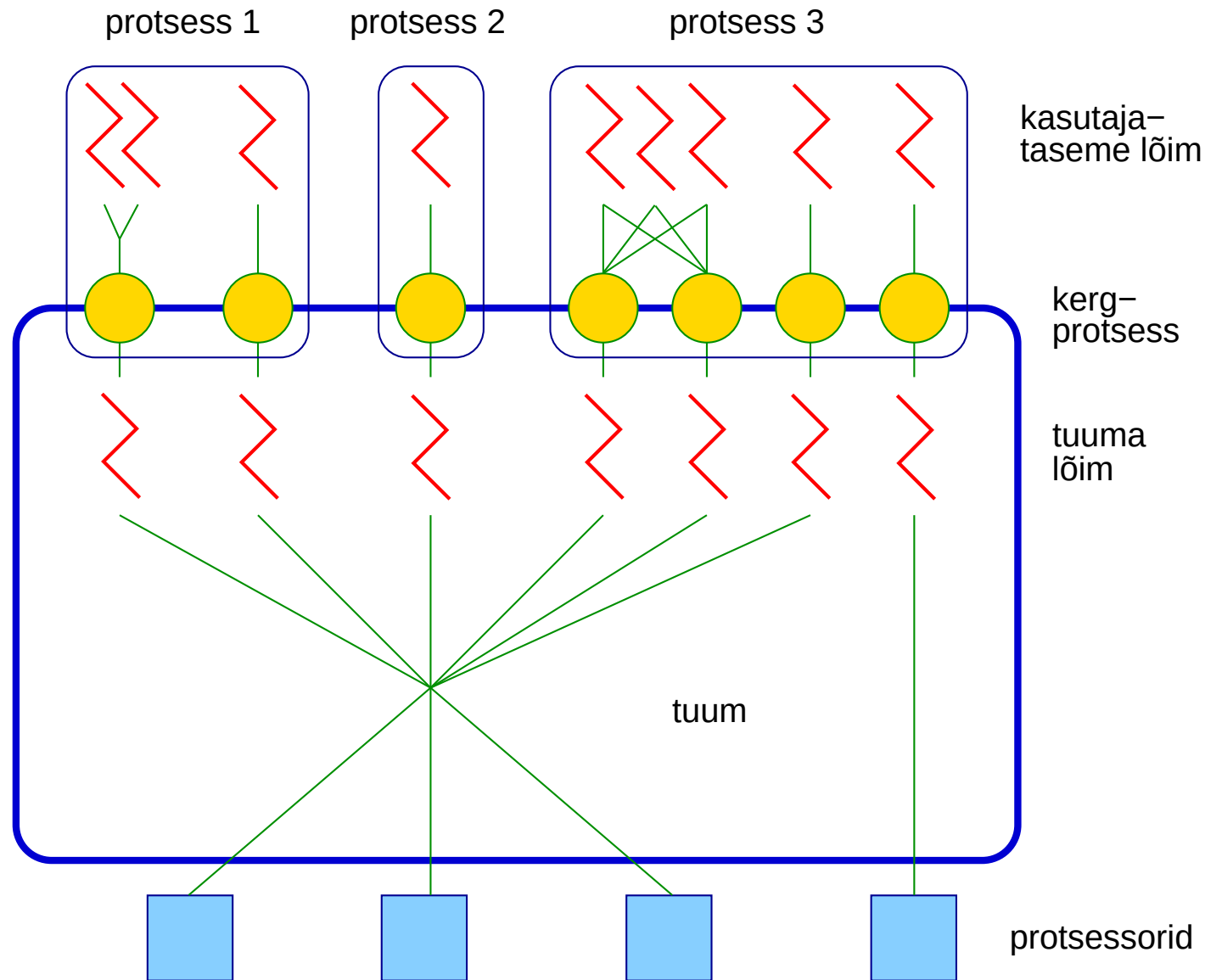
## Pthreads näide

```
int main (int argc, char *argv[])
{
    pthread_t threads[NUM_THREADS];
    int rc, t;
    for (t=0; t<NUM_THREADS; t++) {
        printf("Creating thread %d\n", t);
        rc = pthread_create(&threads[t], NULL, thread_func, (void *)t);
        if (rc) {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(1);
        }
    }
    pthread_exit(NULL);
}
```

## Solarise lõimed

- Läbi ja lõhki lõimeline :)
- LWP (*lightweight process*) — kergprotsess (lõimede liides kasutaja ja tuuma tasemete vahel)
- Igale LWP-le vastab tuuma lõim, tuumas võib lõimesid veel olla
- Kasutajataseme lõimesid saab vajadusel LWP-dega siduda (LWP-sid läheb vaja siis, kui on vaja tuuma teenuseid kasutada)
- Sidumata lõimed jagavad omavahel mingit hulka LWP-sid
- Protsessi poolt kasutatavate LWP-de arvu reguleerib lõimede teek automaatselt (saab ka käsitsi)

# Solarise lõimed





## Windows 2000 lõimed

- Üks-ühele mudel, *ThreadFiber* teegi abil ka mitu-mitmele
- Igal lõimel on ka privaatne andmeala (data, DLL-id jms)
- Kaks magasinini (kasutajataseme ja tuuma taseme jaoks)
- Palju prioriteeditasemeid

## FreeBSD lõimed

- Kuni FreeBSD 4-ni kasutajasemele Pthreads, libc\_r
- Alates FreeBSD 5 on olemas ka tuuma taseme lõimed
- KSE – *Kernel Scheduled Entities*
- N:M teek Pthreads API otsas
- TLS pole hetkel veel valmis, aga ilmselt tuleb

## Linuxi lõimed

- Tuum toetab lõimede tegemist `clone()` abil — luuakse protsessid, mis jagavad rohkem kui `fork()` abil tehtud protsessid (aadressiruum, failipidemed, nimeruumid, . . .)
- `clone()` abil saadud lõimed on nagu protsessid, kontekstivahetus sama, igal oma PID jne.
- Ei eristata protsessi ja lõime, kõik on üks *task*
- Eksisteerib Pthreads'i realisatsioon LinuxThreads (üks-ühele), signaalide töötlemiseks oma lõim
- Alates Linux 2.6 uus implementatsioon — NPTL (*Native POSIX Thread Library*):
  - Tuumale lisati lõimede liides Pthreadsi efektiivseks realiseerimiseks (+TLS)
  - Iga lõim kuulub mingi protsessi alla