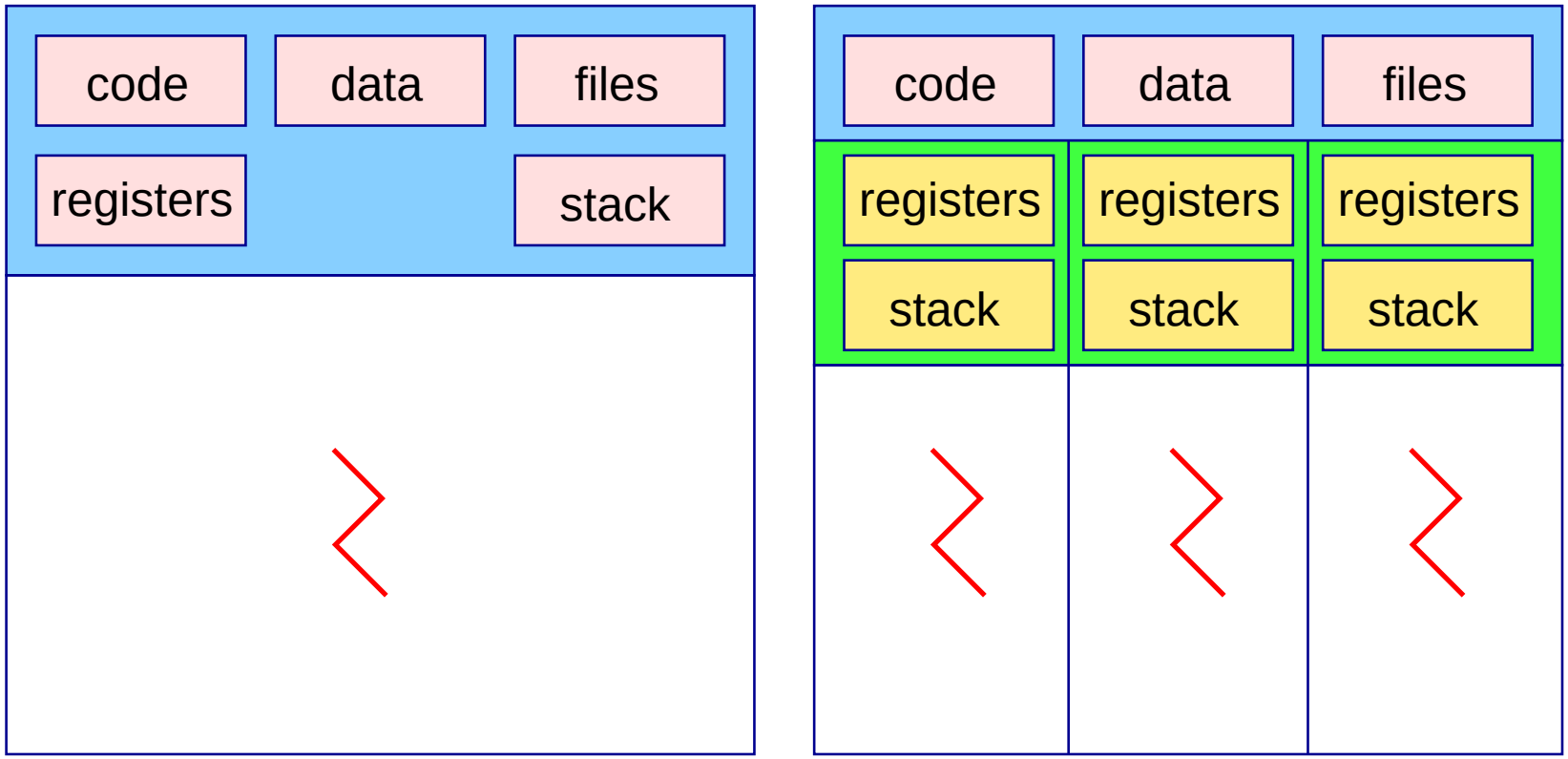
Lõimed

- Lõime mõiste
- Lõimede mudelid
- Probleemid lõimedega seoses
- Pthreads
- Solarise lõimed
- Windowsi lõimed
- FreeBSD lõimed
- Linuxi 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 — aadressiruum on programmi mitme täitja poolt kasutatav
- Kokkuhoid — hoiame kokku jagatud ressursside, uute protsesside loomise ning kontekstivahetuste pealt
- Mitme protsessori ära kasutamine ühe programmi poolt

Kasutaja taseme lõimed

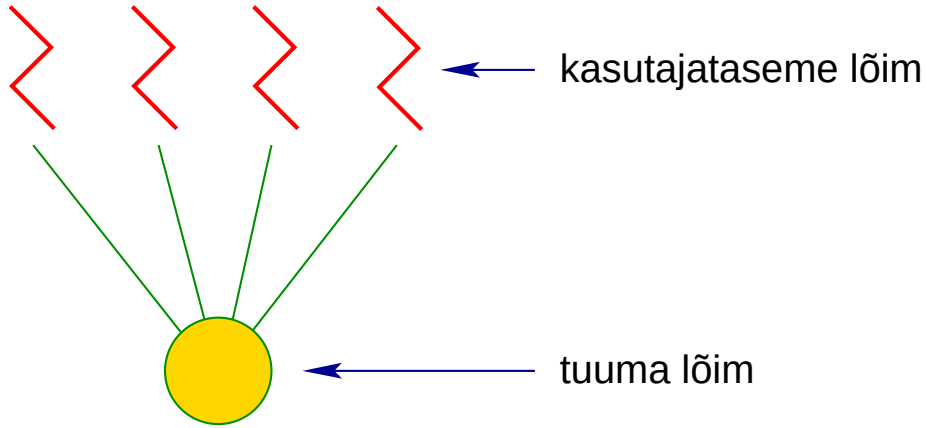
- Lõimede haldus realiseeritakse kasutaja tasemel teegina
- Tuum ei tea midagi, tema jaoks on näha ühte ühelõimelist protsessi
- Haldus on kiire, kuna toimub kasutaja tasemel
- Näiteks:
  - POSIXi Pthreads
  - Mach'i C-threads
  - Solarise lõimed (UI-threads)
  - FreeBSD 4 lõimed
  - Windows NT fiibrid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

Tuuma lõimed

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

Lõimede mudelid

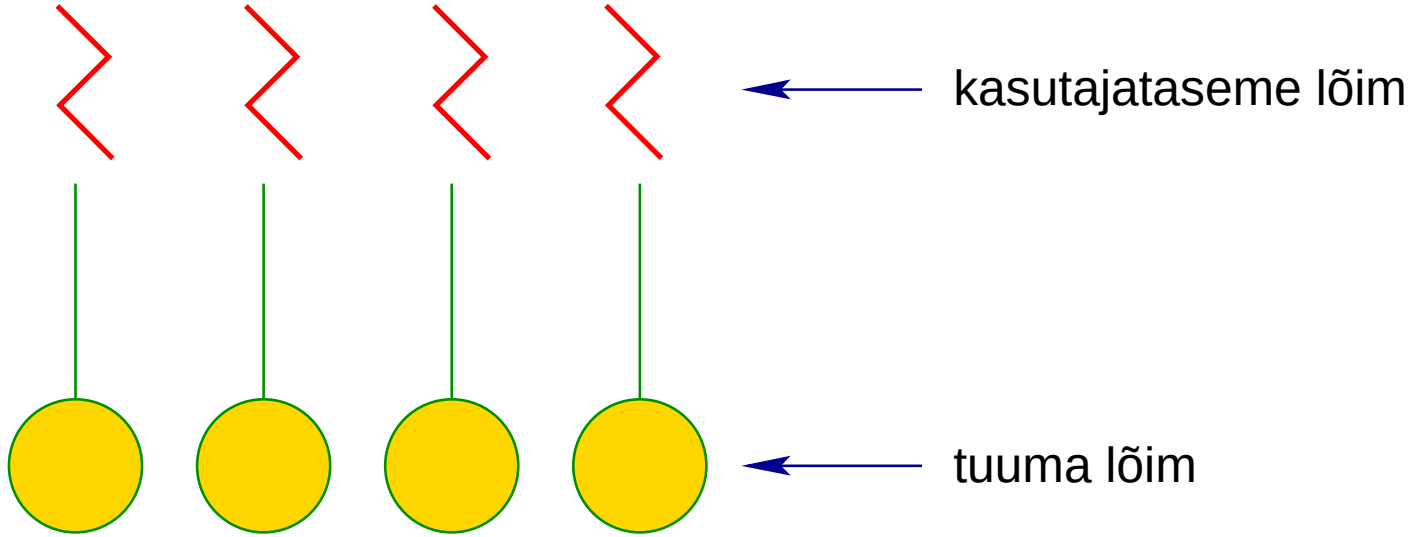
- Mitu ühele — protsessi kõigi kasutaja tasemel lõimede jaoks on üks tuuma lõim



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

Lõimede mudelid

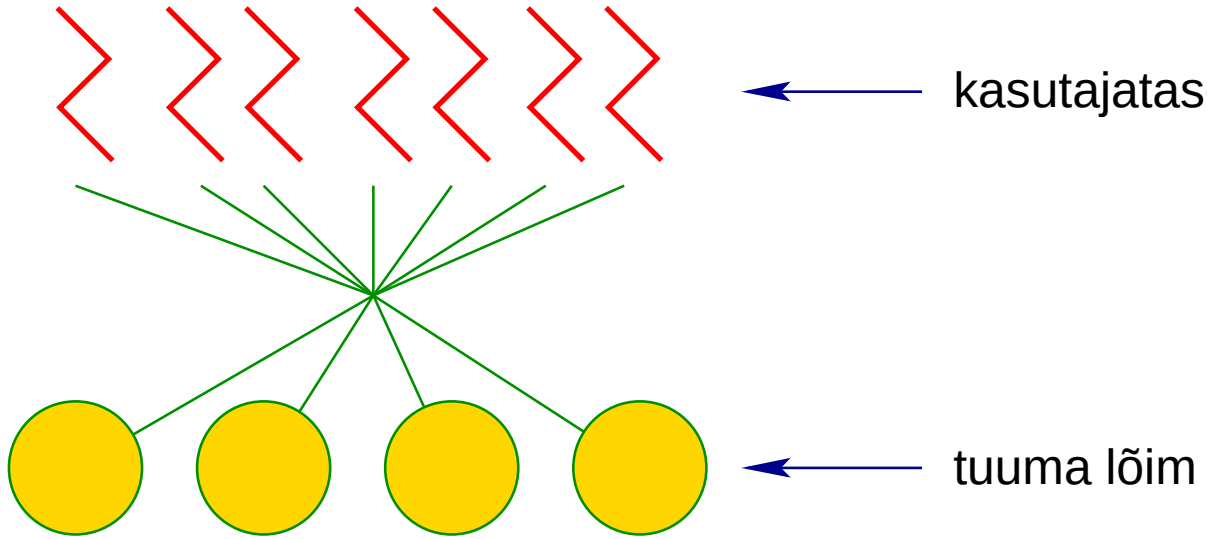
- Üks ühele — igale kasutaja taseme lõimele vastab tuuma lõim



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

Lõimede mudelid

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



  - Kombineerib eelmiste head küljed
  - Keerukam
- *Scheduler activations*

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 ehk POSIX 1c)
- Defineerib programmeerimisliidese lõimedele (C tasemel)
- Kasutaja tasemel, tuuma lõimedega seotud ainult konkreetse implementatsiooni detailide kaudu
- pthread.h, pthread_* funktsioonid
- *De facto* standard Unixites
- Signaalitöötlus on kohmakas, muidu normaalne madala taseme 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);
}
```

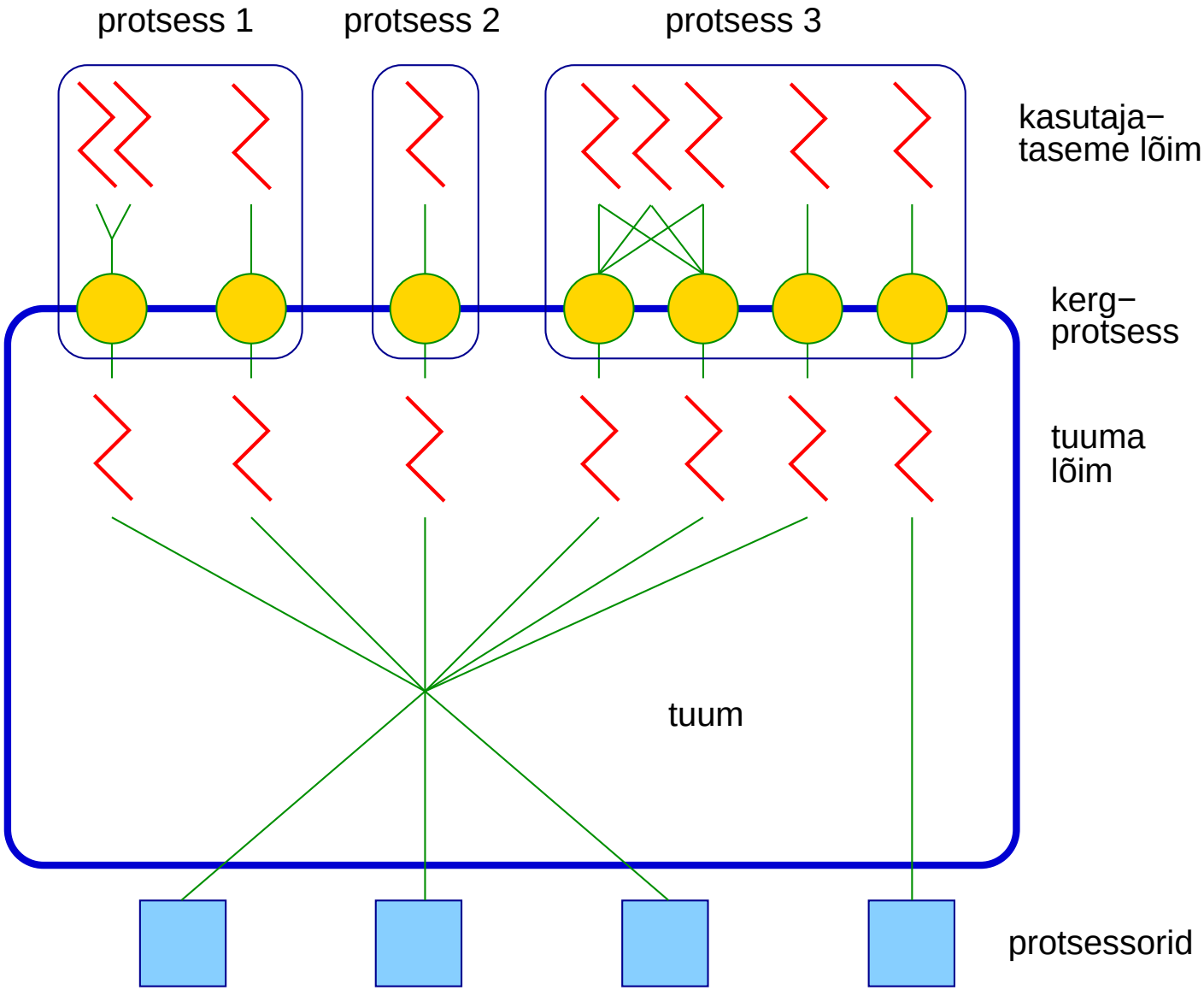
Pthreads näide

```
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
- Kasutaja taseme 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



Windowsi lõimed

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

FreeBSD lõimed

- Kuni FreeBSD 4-ni kasutaja tasemele Pthreads, libc_r
- Alates FreeBSD 5 on olemas ka tuuma taseme lõimed
- KSE – *Kernel Scheduled Entities*
- Esialgu N:M teek Pthreads API otsas
- Tänapäeval 1:1 mudel
- TLS

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