CRUX

A lightweight, i686-optimized Linux distribution

Release 2.0

chris commenda

Linux User Group Linz

Table of Contents

Goals for a New Linux Distro

"make it simple as possible, but not too simple.."
(A. Einstein)

CRUX design:

- **leightweight:**
  whole distribution fits into an iso-image of 212MB
- **optimized:** initially i686/PowerPC optimized
- **simple:** tar.gz-based package system *(slackware)*
- **small:** collection of trimmed packages
- **source based:** complete rebuild possible *(gentoo)*
- **up to date:** ports system Pkgfiles with source location and build instructions *(bsd)*
- **community:** Crux Linux Community (CLC), with searchable packages database, mailing list...
- **new:** Linux features,
  (kernel-2.6.7, gcc-3.3.4, glibc-2.3.3 with nptl, XOrg, AMD64?)
- **extensible:** ports of gnome, kde, openoffice, xine, mplayer, transcode...

Installation Procedure

1. The ISO image is bootable, just insert the CD and reboot your computer

2. Login as root (no password required)

3. Create (if necessary) and format the partition(s) you want CRUX to be installed on.

   $ fdisk /dev/discs/disc?/disc
   $ mkreiserfs /dev/discs/disc?/part?
   $ mkswap /dev/discs/disc?/part?

   special feature: devfs pseudo and reiserfs

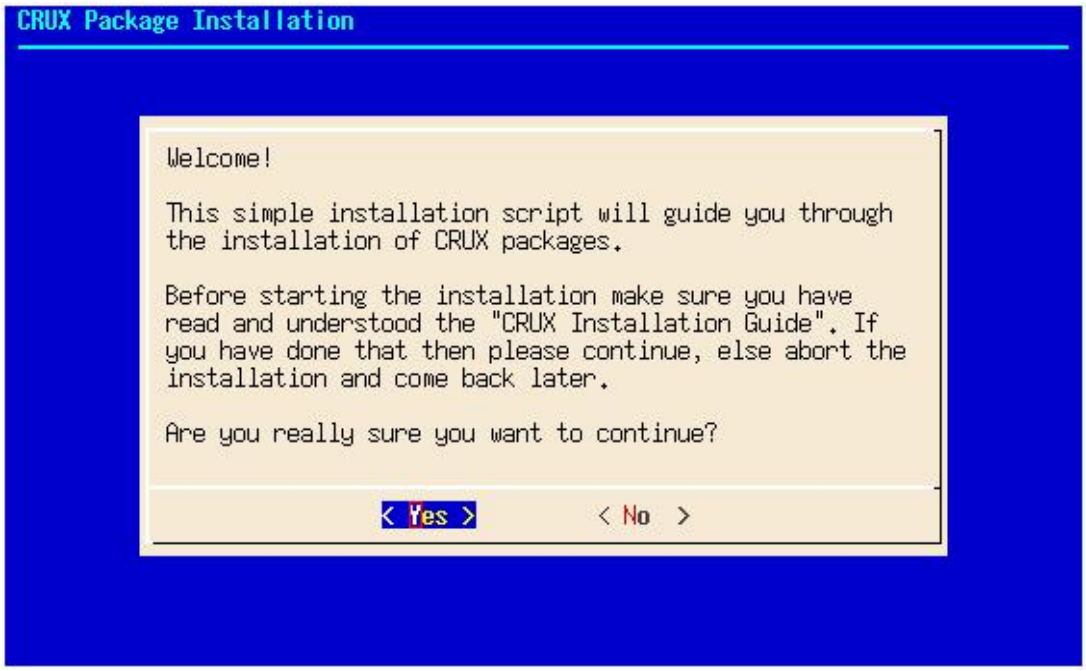
4. Mount the partition on which you want to install this distribution

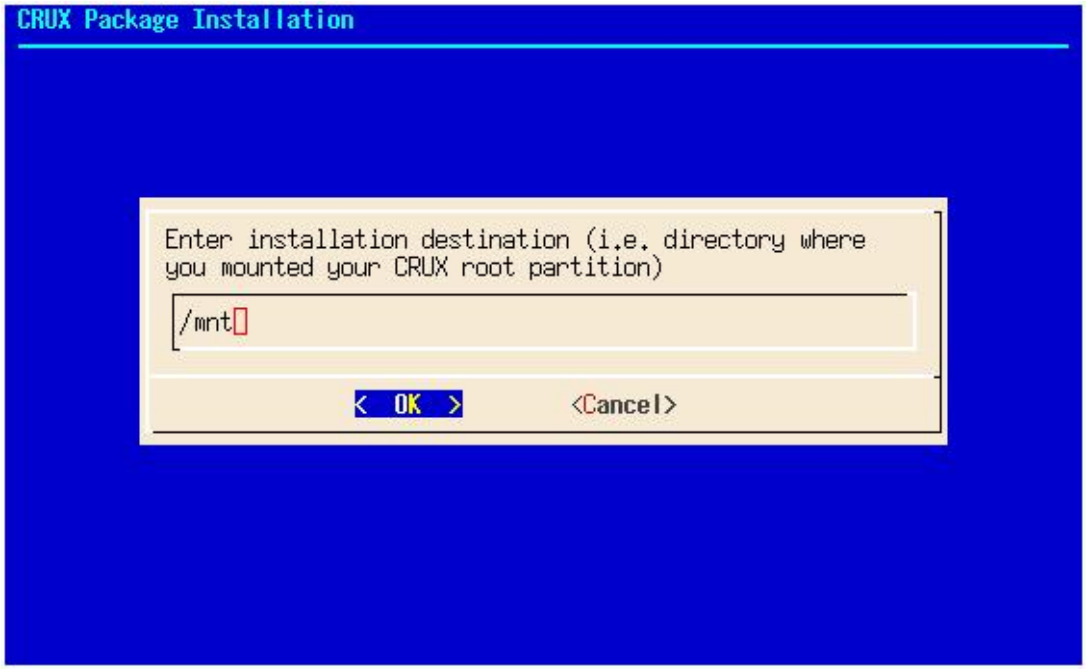
   $ mount /dev/discs/disc?/part? /mnt

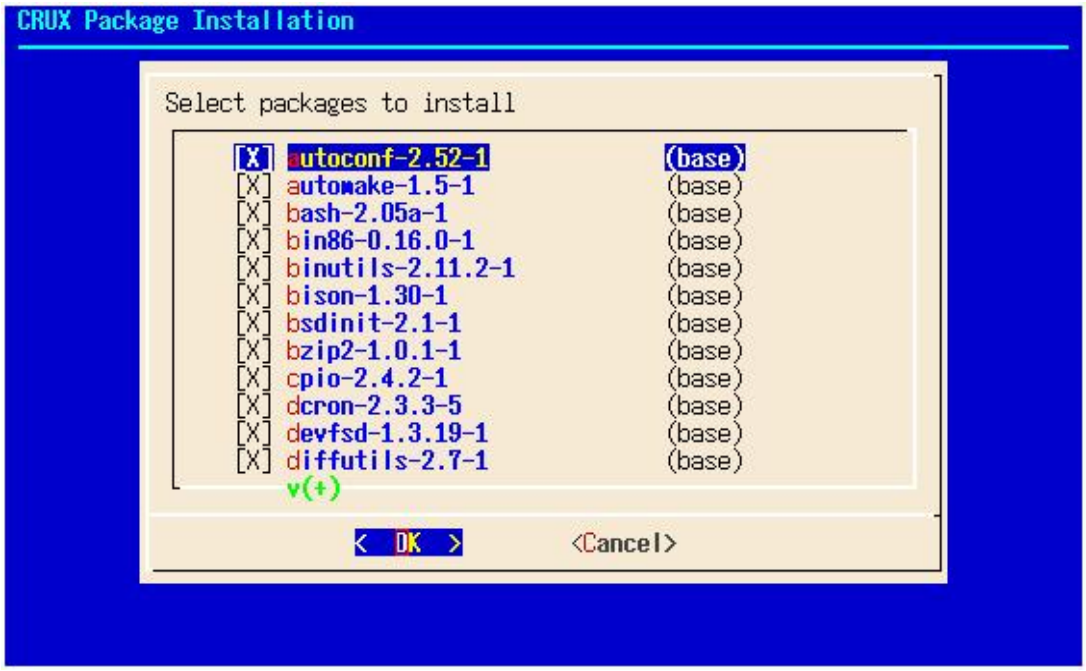
5. Activate your swap partition(s)

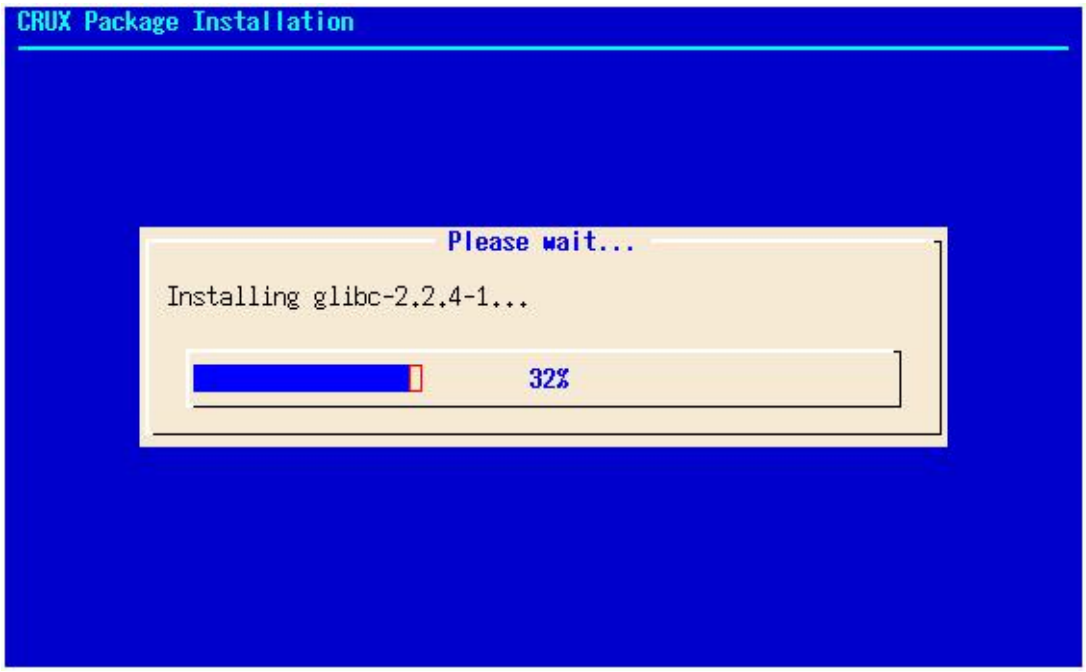
   $ swapon /dev/discs/disc?/part?

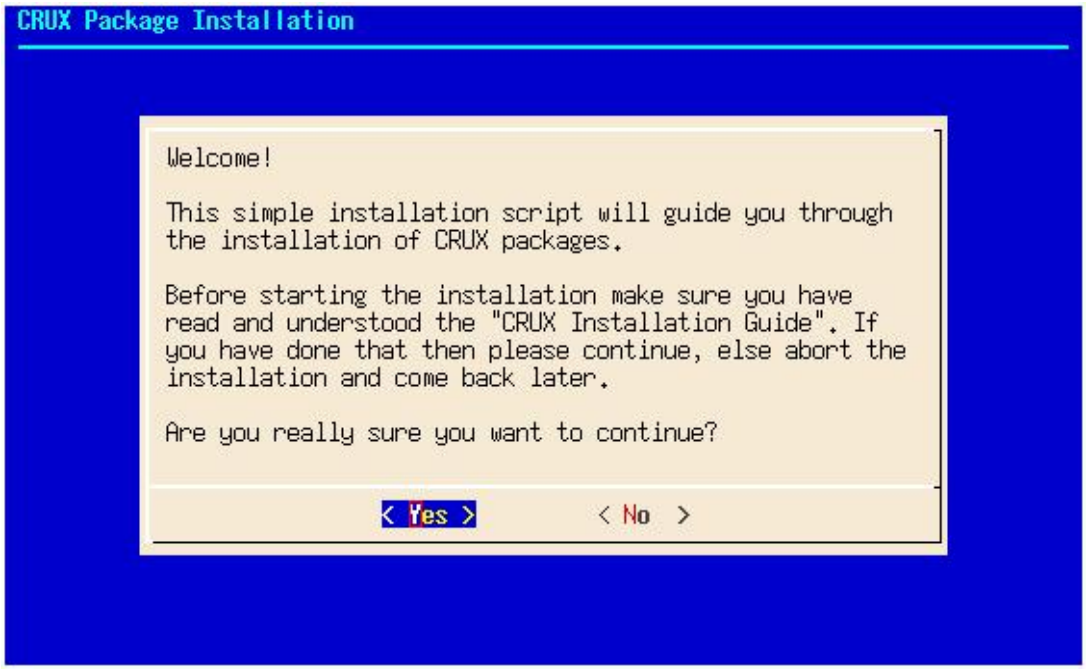
6. Type **setup** to start the package installation script


CRUX Package Installation
Welcome!
This simple installation script will guide you through the installation of CRUX packages.
Before starting the installation make sure you have read and understood the "CRUX Installation Guide". If you have done that then please continue, else abort the installation and come back later.
Are you really sure you want to continue?
< Yes >
< No >


CRUX Package Installation
Enter installation destination (i.e. directory where you mounted your CRUX root partition)
/mnt
< OK >
<Cancel>


CRUX Package Installation
Select packages to install
[X] autoconf-2.52-1 (base)
[X] automake-1.5-1 (base)
[X] bash-2.05a-1 (base)
[X] bin86-0.16.0-1 (base)
[X] binutils-2.11.2-1 (base)
[X] bison-1.30-1 (base)
[X] bsdinit-2.1-1 (base)
[X] bzip2-1.0.1-1 (base)
[X] cpio-2.4.2-1 (base)
[X] dcron-2.3.3-5 (base)
[X] devfsd-1.3.19-1 (base)
[X] diffutils-2.7-1 (base)
v(+)
< OK >
<Cancel>


CRUX Package Installation
Please wait...
Installing glibc-2.2.4-1...
32%



7. Now it's time to compile your kernel and do basic system configuration. The kernel compilation requires that you chroot into your new CRUX installation.

   $ mount -t devfs devfs /mnt/dev
   $ mount -t proc proc /mnt/proc
   $ chroot /mnt /bin/bash

8. Edit /etc/fstab to configure your filesystem(s). Editors **vim** and **pico** are available

9. Edit /etc/rc.conf to configure font, keyboard, timezone, hostname and services.

```
# /etc/rc.conf:  system configuration
#
FONT=default
KEYMAP=de
TIMEZONE=UTC
HOSTNAME=seth
SERVICES=(net crond sshd portmap nfs cups)
# End of file
```

10. Edit /etc/rc.d/net, /etc/hosts and /etc/resolv.conf to configure your network (ip-address/gateway/hostname/domain/dns)

```
# /etc/rc.d/net:  start/stop network
#
case $1 in
start)
/sbin/ifconfig lo 127.0.0.1
/sbin/ifconfig eth0 192.1.168.1 netmask 255.255.255.0
#/sbin/route add default gw xxx.xxx.xxx.xxx
;;
stop)
/sbin/ifconfig eth0 down
/sbin/ifconfig lo down
;;
restart)
$0 stop
$0 start
;;
)
echo "usage:  $0 [start|stop|restart]"
;;
esac
# End of file
```

11. Go to /usr/src/linux-2.6.7, configure and compile a new kernel

```
$ cd /usr/src/linux-2.6.7
$ make menuconfig
$ make -j 2 all
$ make modules_install
$ cp arch/i386/boot/bzImage /boot/vmlinuz
$ cp System.map /boot/
```

12. Edit /etc/lilo.conf to boot the kernel you just compiled and run **lilo** to make the new system bootable

13. Remove the CRUX CD-ROM from your drive and reboot from harddisk

Package System

- Installing a package is done by using

  $ pkgadd bash#2.05-1.pkg.tar.gz

- Upgrading a package

  $ pkgadd -u bash#2.05-1.pkg.tar.gz

- Removing a package is done by using

  $ pkgrm bash

- Querying the package database is done using **pkginfo**

  $ pkginfo -i

```
autoconf 2.52-1
automake 1.5-1
<...>
xmms 1.2.7-1
zip 2.3-1
zlib 1.1.4-1
```

  $ pkginfo -l bash

```
bin/bash
bin/sh
etc/
etc/profile
usr/
usr/man/
usr/man/man1/
usr/man/man1/bash.1.gz
usr/man/man1/sh.1.gz
```

$ pkginfo -o bin/ls

```
e2fsprogs usr/bin/lsattr
fileutils bin/ls
modutils sbin/lsmod
```

- Creation of a package is done by **pkgmk** using a `Pkgfile`

```
# Description:  LAM (Local Area Multicomputer)
# an MPI programming environment
# URL: http://www.lam-mpi.org
# Maintainer:  Chris Commenda <chris.commenda@gmx.net>
# Group:  net

name=lam
version=7.0.6
release=1
source=(http://www.lam-mpi.org/download/$name-$version.tar.bz2)

build(){
cd $name-$version
./configure --prefix=/usr --sysconfdir=/etc --with-exceptions
--without-fc
make -j 2
make DESTDIR=$PKG install
rm -rf $PKG/usr/share
}
```

- Package guidelines:
  - install in specific directories
  - remove junk files (info pages, online documentation, Files related to NLS (national language support)

Ports System

The use of the word port in this context is borrowed from the BSD world. CRUX users use the ports utility to download ports from the CVS repositorty and place them in `/usr/ports/`. The ports utility uses CVSup or httpup (behind firewall) to do the actual downloading and synchronization.

- bring your local ports structure up to date

  $ ports -u
  ```
  Connected to cvsup.fukt.bth.se
  Updating collection base/cvs
  ...
  Updating collection opt/cvs
  ...
  Finished successfully
  ```

- listing local ports

  $ ports -l
  ```
  base/autoconf
  base/automake
  base/bash
  base/bin86
  base/binutils
  base/bison
  ...
  ```

- listing version differences

  $ ports -d
  ```
  Collection Name Port Installed
  base glibc 2.3.2-2 2.3.3-1
  opt gtk 2.2.0-1 2.4.0-1
  ```

- download, build and upgrade

  $ cd /usr/ports/my/lam
  $ pkgmk -d -u

Native POSIX Thread Library (NPTL)

1. **What is a thread ?**

   A thread–sometimes called an execution context or a lightweight process–is a single sequential flow of control within a program. You use threads to isolate tasks. Each operation in a thread runs independently from the operations in the others threads, but at the same time.

2. **First Implementation of Linux Thread Library (1996)**

   - each *user-level thread* is handled by *one kernel thread*
   - *no use of registers* (thread local memory located using fixed relationships between stack pointer and position of thread descriptor)
   - no synchronization primitives; *fragile signals used* instead by the kernel
   - *manager thread* handels signals, thread creation, process management...

3. **Improvements over time**

   - *use of thread registers*
     location of thread-local data is no longer a time consuming operation. Brought more speed and flexibility, but restricted number of threads (IA32 - register starved, 8192 threads)
   - improvement of kernel (2.4.xx), by using segment registers and improved creation of kernel threads

4. **Goals for New Implementation**

   - POSIX standard compliance
   - effective use of Symmetric Multi Processing (SMP)
   - Low startup and link-in costs
   - binary compatibility
   - scalability
   - machine architecture support
   - integration with C++ (exception handling)

5. **Design Decisions**

   - 1-on-1 (one user-level thread for on kernel thread)
   - avoid manager thread
   - list of all threads still necessary
   - new functionality added to kernel to implement synchronization
   - optimization of memory allocation for fast startup

6. Performance Results

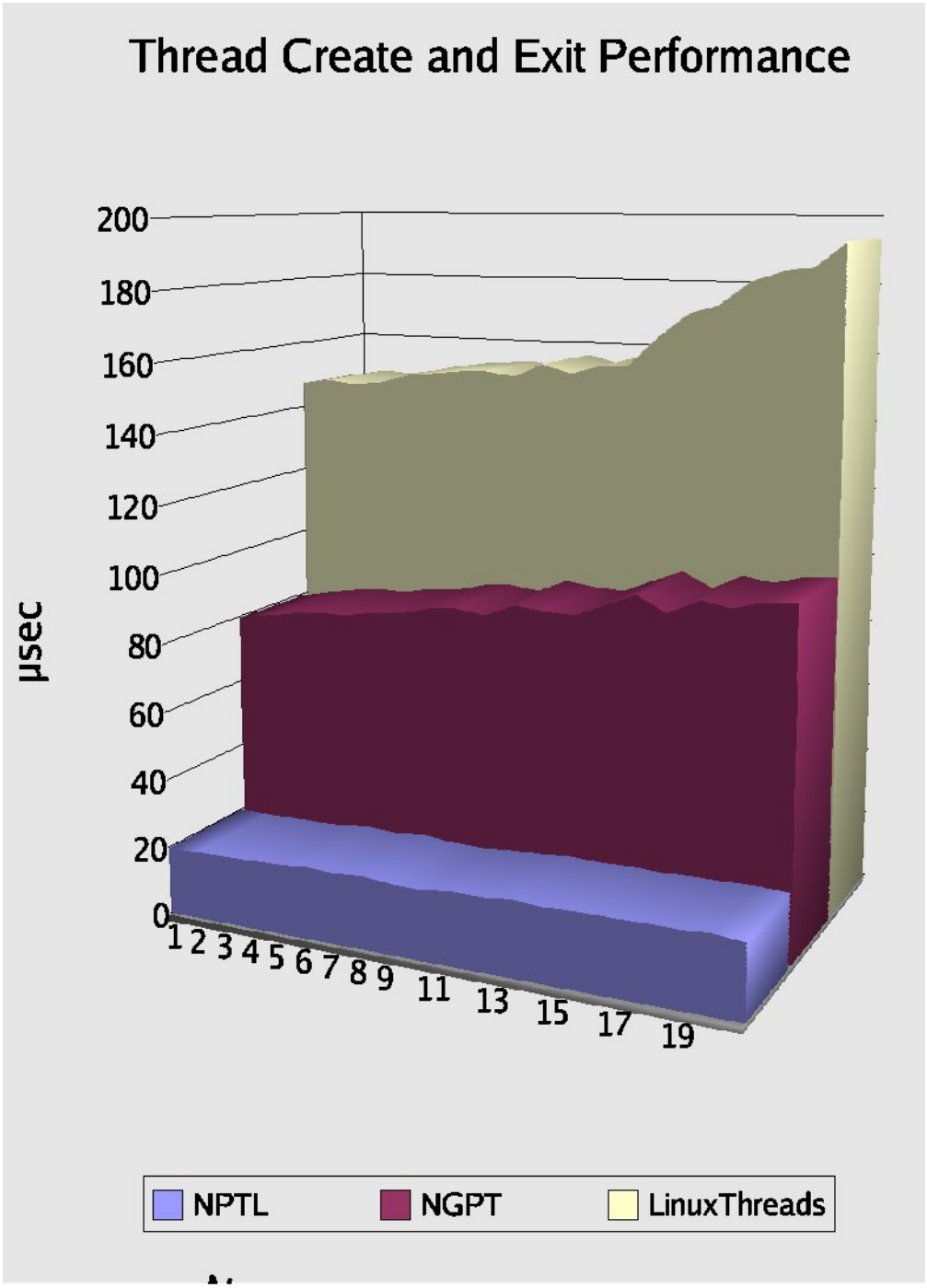


Figure 1: Varying number of Toplevel Threads

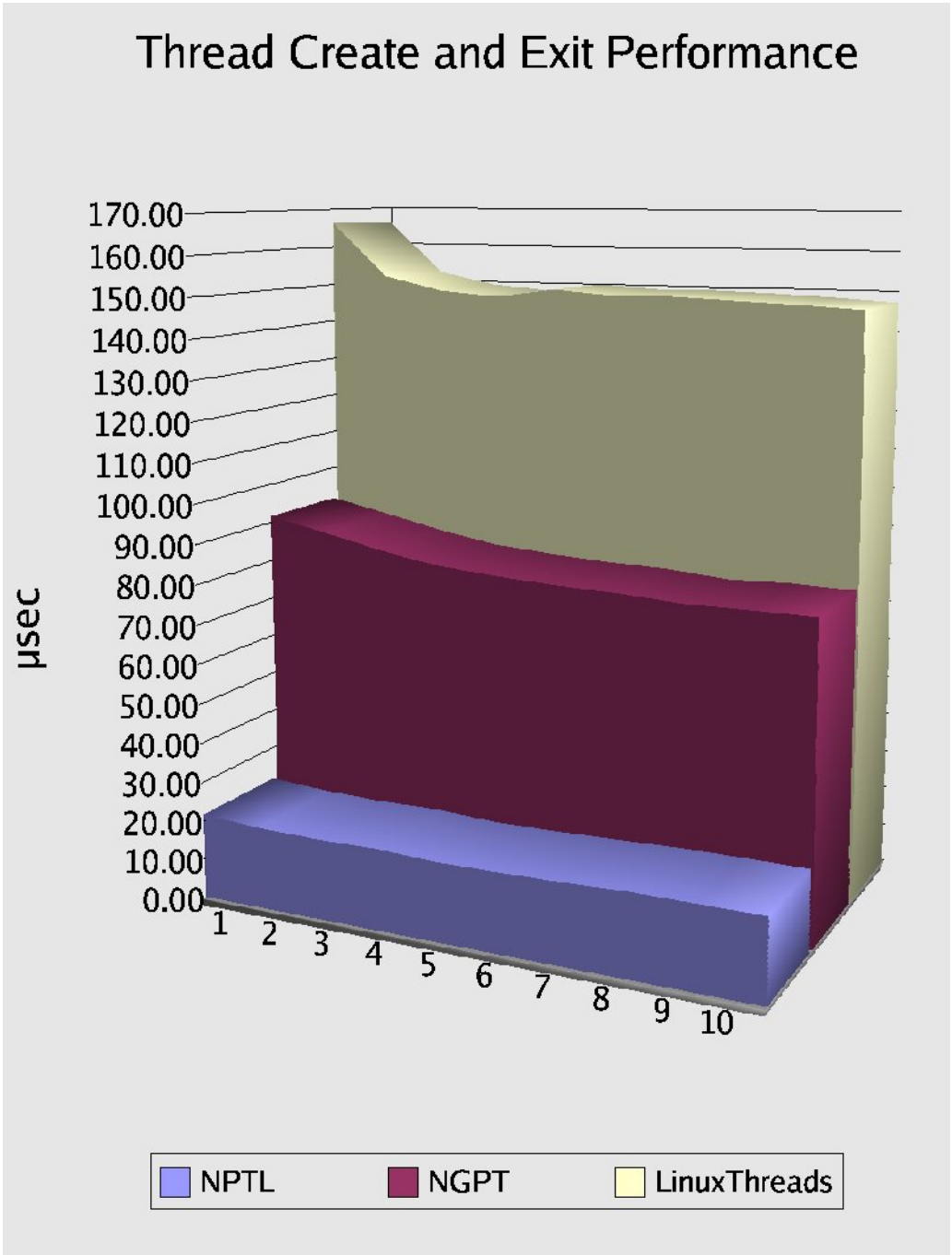


Figure 2: Varying number of Concurrent Children

Program **csfast5a** will be assigned to start any number of threads, where each thread locks and unlocks (using a process synchronization primitive) a given number of times, and when done, records the time it took.

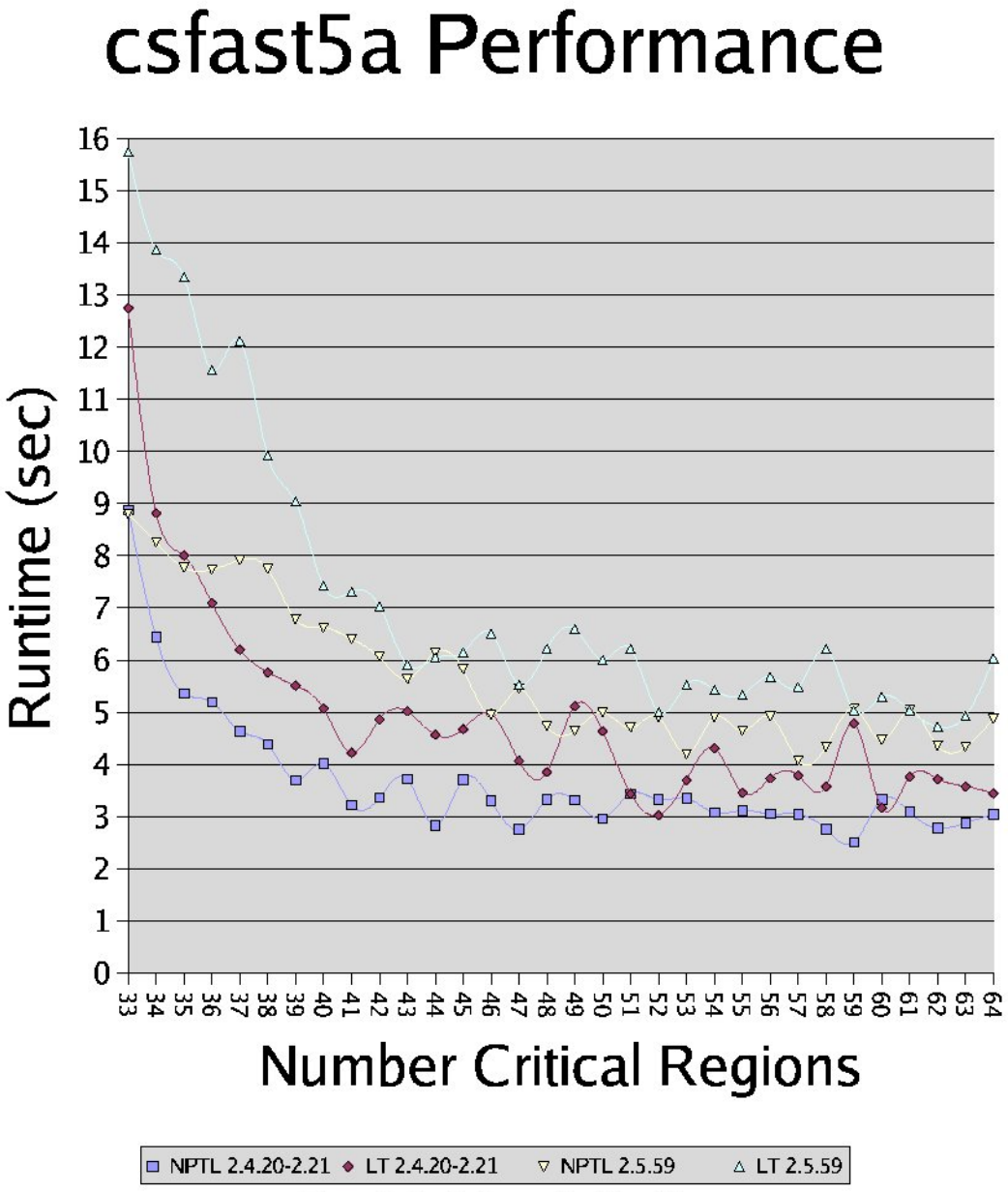


Figure 3: Lock Contention Handling

The Bibliography

1. http://www.crux.nu

2. http://people.redhat.com/drepper/nptl-design.pdf

3. http://www-106.ibm.com/developerworks/linux/library/l-rt10/index.html?t=gr,lnxw01=ConSwiP2