COLINUX INSTALLATION

Editor:
Tomáš Mandys, tomas.mandys@2p.cz (2p plus)

Home site:
http://www.2p.cz

Document status:

Version 1.0	First release
Version 1.1	update for coLinux 0.6.2

Table of Contents

Disclaimer

The information of this document is provided ,“AS IS‘, with no warranties whatsoever, excluding in particular any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. This document is provided for information purposes only.

1. Introduction

This document describes how to install coLinux on a machine. On the machine will be installed two systems, Windows and Linux that may be booted directly from boot manager. In addition virtual coLinux that starts from Windows will boot from Linux partition.

2. Installation

2.1. Native Linux

Install a Linux system like Fedora to a dedicated partition. Choose *GRUB* installation to boot record or to MBR. If *GRUB* resides in boot record then it is necessary to configure Windows `boot.ini` to support multi-boot.

Windows boot manager requires than boot records for another systems resides on a Windows disk.

Reboot Linux just after installation from install CD and go to rescue mode `linux rescue`. Let installer find Linux system and copy first sector of Linux boot partition to Windows disk `C:`.

1. prepare a mount point, e.g. `mkdir /mnt/c`
2. mount Windows `C:` disk `mount /dev/hda2 /mnt/c`
3. copy first sector `dd if=/dev/hda3 of=/mnt/c/linux.ldr bs=512 count=1`
4. edit `/mnt/c/boot.ini`, add line `C:\linux.ldr`

Note that Fedora core 2 does not support NTFS nor in kernel nor as rpm in installation CD.

To support NTFS download rpm package from http://linux-ntfs.sourceforge.net/rpm/instructions.html. It's necessary to download rpm corresponding to current kernel version (`uname -r`), install rpm and exec `modprobe ntfs`.

Boot to native Linux and create virtual device nodes

```
mknod /dev/cobd0 b 117 0
mknod /dev/cobd1 b 117 1
mknod /dev/cobd2 b 117 2
...
mknod /dev/cobd7 b 117 7
# max. 32 nodes in 0.6.3
```

Install coLinux kernel modules from tarball to `/lib/modules/2.4.26-co-0.6.1`

2.2. Windows

Reboot to Windows and install coLinux.

Make shortcut that will start virtual coLinux machine

```
"C:\Program Files\coLinux\colinux-daemon.exe" -c default.colinux.xml
```

Now prepare coLinux configuration in `default.colinux.xml`

1. Add floppy support

```
<block_device index="2" path="\Device\Floppy0" enabled="true">
</block_device>
```

2. Add partitions. Note that extended partition is not counted. `/dev/hda7` is equal to `Partition6` in XML

```
<block_device index="7" path="\Device\Harddisk0\Partition6" enabled="true">
</block_device>
```

3. Add root partition to `bootparams`

```
<bootparams>root=/dev/cobd7</bootparams>
```

4. Add CD-ROM support, but this does not work on my machine, I'm using CDROM using samba (see `fstab`)

```
<block_device index="3" path="\\Device\CdRom0" enabled="true">
```

It's probably not possible access native Windows partitions because are locked by system. Windows partitions can be accessed from coLinux using samba client or via `cofs` (>=0.6.2).

It seems that when virtual coLinux booting RH7, RH9 (Fedora) is unable `fsck` virtual partitions. To bypass the file system checks add `bootparams` like those:

```
<bootparams>root=/dev/cobd7 fastboot nogui</bootparams>
```

5. Add access to native Windows partitions via `cofs`. It seems that it’s problem with national char set support.

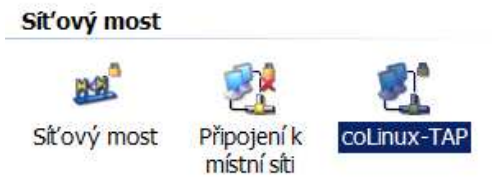
```
<cofs_device index="0" type="flat" path="\??\d:\" enabled=true />
```

Networking

coLinux support bridged networking or NAT. I’m using bridged networking that requires three addresses. Note that bridged networking does not work using WiFi card as according a report many wireless cards do not support non-own MAC addresses.

1. real NIC (old Windows address IP), e.g. `192.168.1.20`
2. colinux IP, e.g. `192.160.1.29`
3. *TAP-Adapter* IP, if IP pool is limited it can be used a dummy address from LAN IP range, e.g. `192.168.1.22`

Install *TAP-Win32 adapter* driver from *OemWin2k.inf* unless already installed.



Setup Windows networking

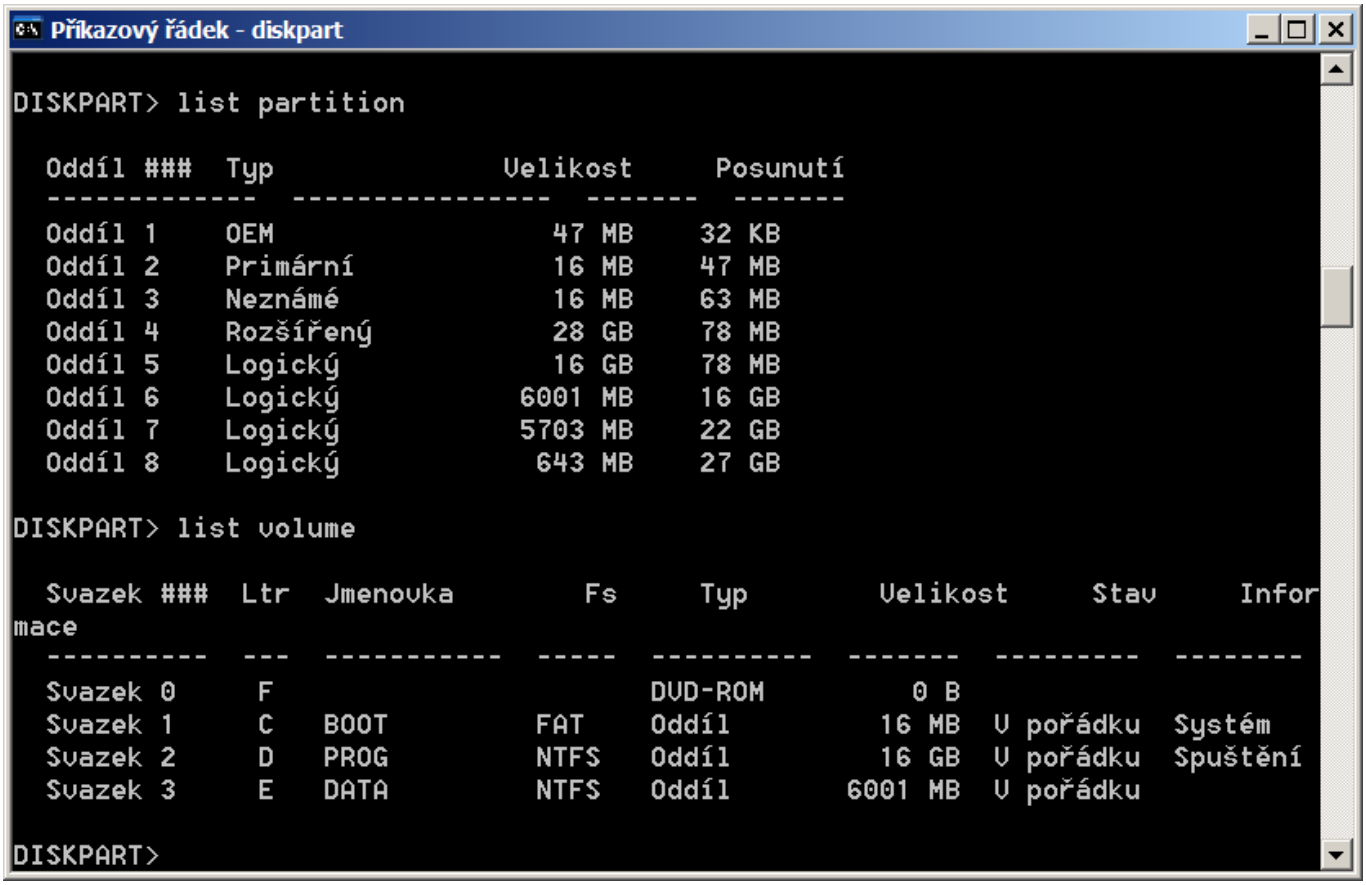
1. real NIC (no change)
2. TAP-Adapter, set IP and mask; leave gateway and DNS empty
3. select real NIC and TAP-Adapter, right click and do *"Bridged connection".* New bridge should appear. Setup the same IP properties as in real NIC (IP,mask,gateway,DNS)

Add in configuration file reference to TAP adapter

```
<network index="0" type="tap" name="coLinux-TAP"></network>
```

Edit windows`/system/drivers/etc/host` and `lmhosts`

Example




```
<?xml version="1.0" encoding="UTF-8"?>
<colinux>
    <block_device index="0" path="\Device\Floppy0" enabled="true">
    </block_device>
    <block_device index="2" path="\Device\Harddisk0\Partition2" enabled="true">
    </block_device>
```

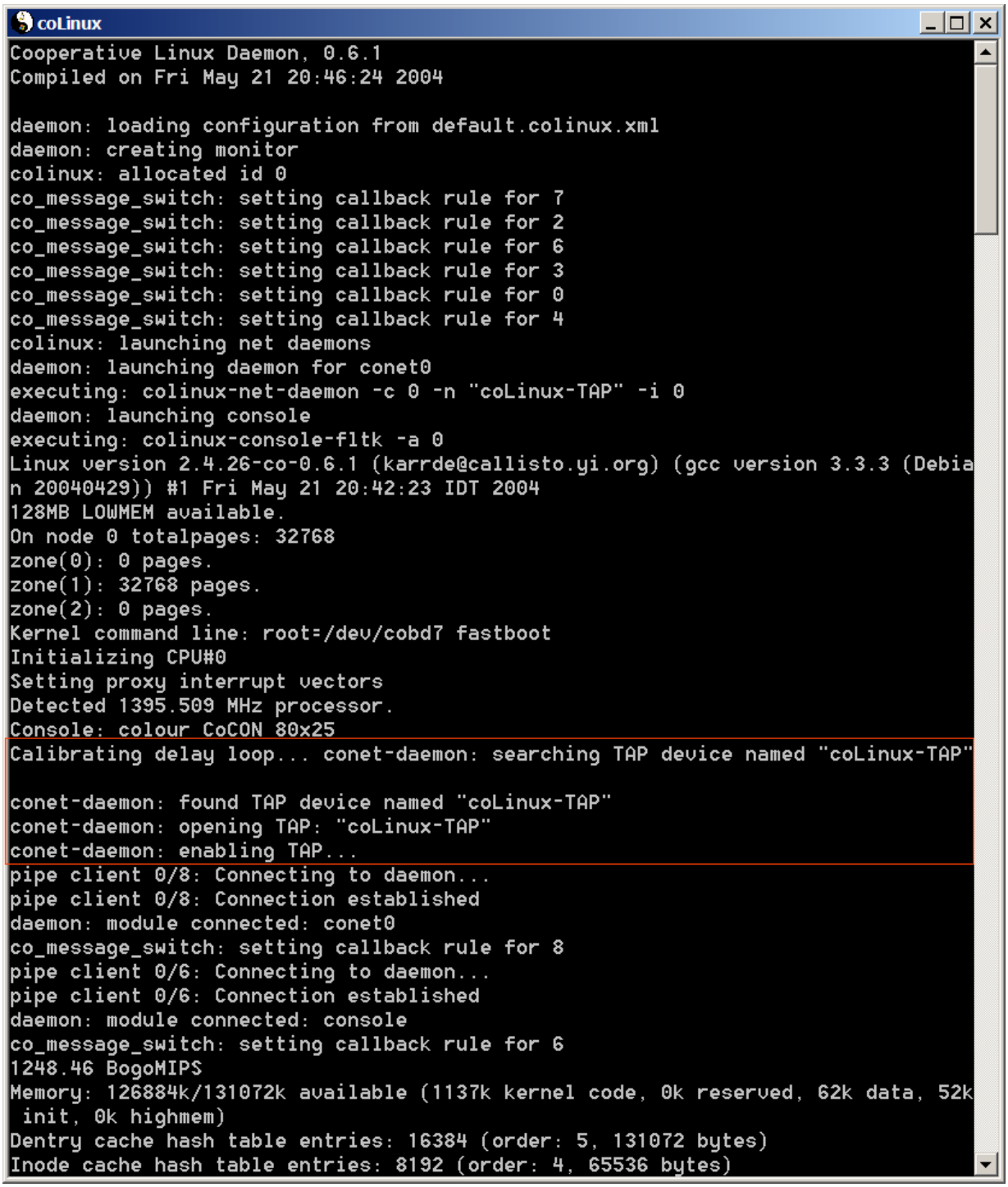
```
    <block_device index="5" path="\DosDevices\D:\" enabled="true">
    </block_device>
    <block_device index="6" path="\DosDevices\E:\" enabled="true">
    </block_device>
    <block_device index="7" path="\Device\Harddisk0\Partition6" enabled="true">
    </block_device>
    <block_device index="8" path="\Device\Harddisk0\Partition7" enabled="true">
    </block_device>
    <block_device index="9" path="\Device\Cdrom0" enabled="true">
    </block_device>

    <cofs_device index="0" type="flat" path="\??\d:\" enabled=true />
    <cofs_device index="1" type="flat" path="\??\e:\" enabled=true />

    <bootparams>root=/dev/cobd7 fastboot</bootparams>
<!--    <bootparams>root=/dev/cobd7 append="hda=255,63,7476"</bootparams> -->
    <image path="vmlinux"></image>
    <memory size="128"></memory>
    <network index="0" type="tap" name="coLinux-TAP"></network>
</colinux>
```

2.3. coLinux

Start coLinux virtual machine. Look if TAP adapter started correctly.



Networking

Create special configuration for coLinux virtual machine. Note that RH9 (Fedora) requires configurations in special directory (`profile`)

```
cd /etc/sysconfig/network-scripts
mkdir profile
cp ifcfg-eth0 profile/ifcfg-eth0.linux
cp ifcfg-eth0 profile/ifcfg-eth0.colinux
```

Edit `profile/ifcfg-eth0.colinux`, set *IPADDR* to coLinux IP (`192.168.1.22`) and *GATEWAY* to *TAP-Adapter IP* (`192.168.1.29`) (or to real gateway ???)

Add DNS servers to `/etc/resolv.conf` (the same for Linux and coLinux)

Edit `/etc/host`

Mounting

It's possible to have multi mount points in `/etc/fstab` like `/dev/hda7` and `/dev/cobd7` in one file but I prefer creating two separate files.

```
cd /etc
cp fstab fstab.linux
cp fstab fstab.colinux
```

Edit `fstab.colinux`

```
/dev/cobd7    /             ext2    defaults,errors=remount-ro 0 0
#/dev/cobd0   /mnt/floppy   auto    noauto,owner    0 0
none          /proc         proc    defaults        0 0
none          /dev/pts      devpts  gid=5,mode=620  0 0

/dev/cobd8    swap          swap    defaults,mode=660   0 0
#/dev/cobd2   /boot         ext2    defaults        0 0
//kudu/Prog   /mnt/d        smbfs   username=xxx,password=**
//kudu/Data   /mnt/e        smbfs   username=xxx,password=**

//kudu/cdrom  /mnt/cdrom    smbfs   username=xxx,password=**,fmask=666,dmask=777,rw

cofs0         /mnt/d        cofs    defaults        0 0
cofs1         /mnt/e        cofs    defaults        0 0
```

Profiles

Prepare manipulation script that detects if Linux or coLinux is booting and adjust configuration.

Create `/etc/rc.d/rc.colinux` script

```
#!/bin/bash
# colinux profiling
if uname -r | grep "co-" > /dev/nul ; then
  HW_PROFILE=colinux
else
  HW_PROFILE=linux
fi
export HW_PROFILE

echo "Checking machine [ ${HW_PROFILE} ]"
# there are /rootfs / and  /dev/root /
ROOT_RO=`grep " / " /proc/mounts | awk '{ print $4 }' | grep "ro"`

if [ "${ROOT_RO}" ] ; then
  /bin/mount -o remount,rw /
fi

/bin/rm /etc/fstab
```

```
/bin/rm /etc/sysconfig/network-scripts/ifcfg-eth0

ln -s /etc/fstab.${HW_PROFILE} /etc/fstab
ln -s /etc/sysconfig/network-scripts/ifcfg-eth0.${HW_PROFILE} /etc/sysconfig/network-
scripts/ifcfg-eth0

if [ "${ROOT_RO}" ] ; then
  /bin/mount -o remount,ro /
fi
```

Call profile script from `/etc/rc.d/rc.sysini`

```
...
# Mount /proc (done here so volume labels can work with fsck)
action $"Mounting proc filesystem: " mount -n -t proc /proc /proc

# colinux hardware profile
/etc/rc.d/rc.colinux
...
```

Finalization

Note if a NumLock manipulation (`setled`) is performed in `/root/.bash_profile` then it brings NumLock problems. NumLock has not the same state for Windows applications and in coLinux console.

Note that HW_PROFILE environment variable set in `/etc/rc.d/rc.colinux` disappeared.

Reboot coLinux and test networking. Each address except TAP-Adapter should be pingable.

Samba

Configure Windows to provide disk for sharing, set a user that has enabled remote connection, enable file sharing at firewall (port 139). Note that for some Windows it's necessary to apply some changes in registry, see *Samba documentation*.

Check Windows resources provided by Windows using

```
smsclient -U <winuser> -L <win_host>
```

Add permanent mounting of Windows drives to `/etc/fstab.colinux`

Configure samba server at colinux to provide colinux drives to Windows and start it.

1. edit `/etc/samba/smb.conf`
2. add user `smbpasswd –a <samba_user> ****`
3. open port 139 at firewall (*iptables*, *ipchains*): rule: `-A input -s 192.168.1.0/24 53 -d 0/0 -p -tcp -j accept`
4. start Samba server `/etc/rc.d/init.d/smb start`
5. put link to `/etc/rc.d/rc3.d` to start Samba server automatically
6. Check connection from Windows `net use s: \\colinux\user /u:samba_user`

Console

Colinux machine may be controlled via `fltk` or `nt` console (`colinux-console-xxx.exe`). Select required console via `-t xxx` command line parameter. Size of console it's hardcoded in exec and scrolling does not work.

`Putty` console may be used too, of course, but networking must be working to connect virtual machine. It brings console scrolling via (`Shift+PgUp/Down`).

coLinux as NT service

1. colinux-daemon.exe" –install-driver
2. colinux-daemon.exe" -c default.colinux.xml –install-service

coLinux Manager

A tiny GNU utility written in .NET C# resides at taskbar and helps starting coLinux service or associated console. Do not forget set console in Settings menu and restart utility! See http://www.biermana.org/

2.4. X-Windows

CoLinux does not support any X-Windows server. To open X-Windows application it's necessary install a 3rd party X-Windows server for Windows.

I have found following X-Servers:

1. *CygWIN* (http://www.cygwin.com), free open source server requires huge *CygWIN* environment installation
2. commenrcial http://www.starnet.com X-Windows server *X-Win32* (19MB, approx.200USD per license)
3. *NX server/client* (http://www.nomachine.com), commercial server, partially as open source

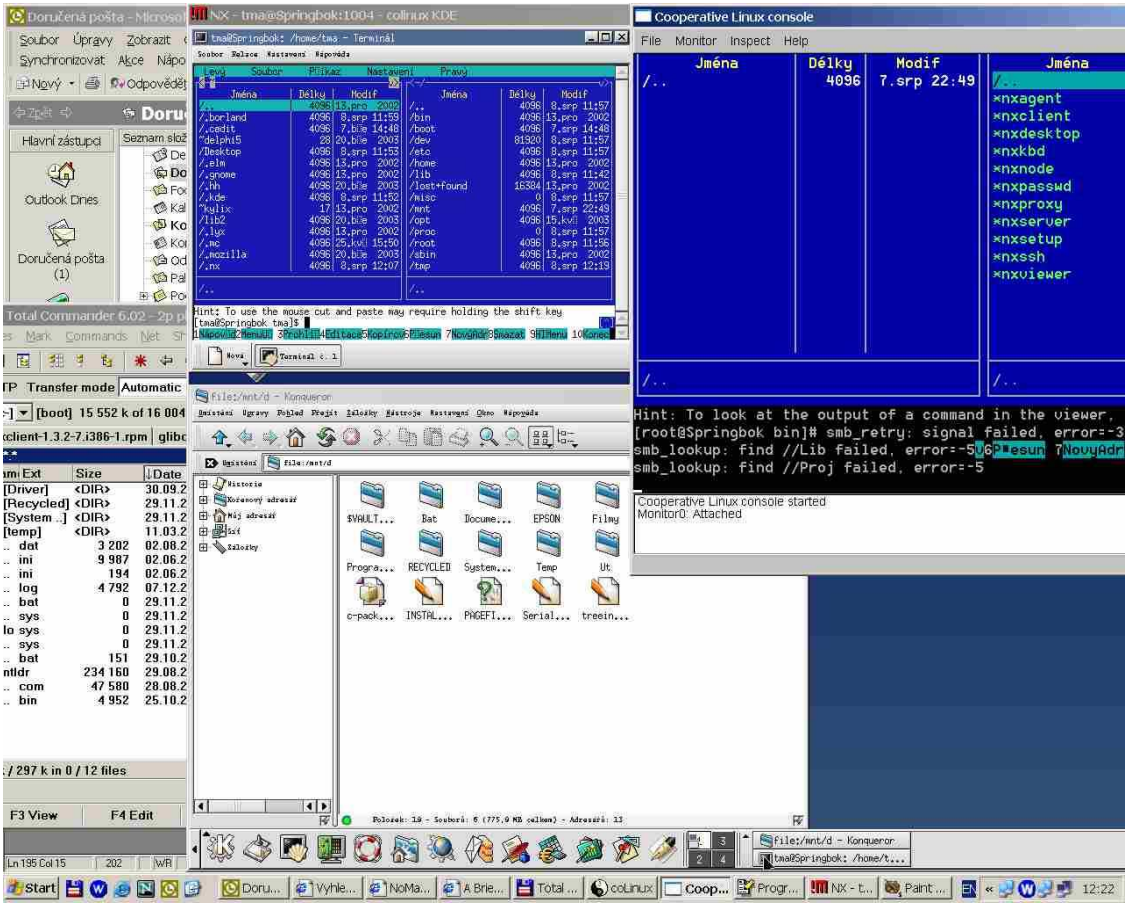
XDeep/32

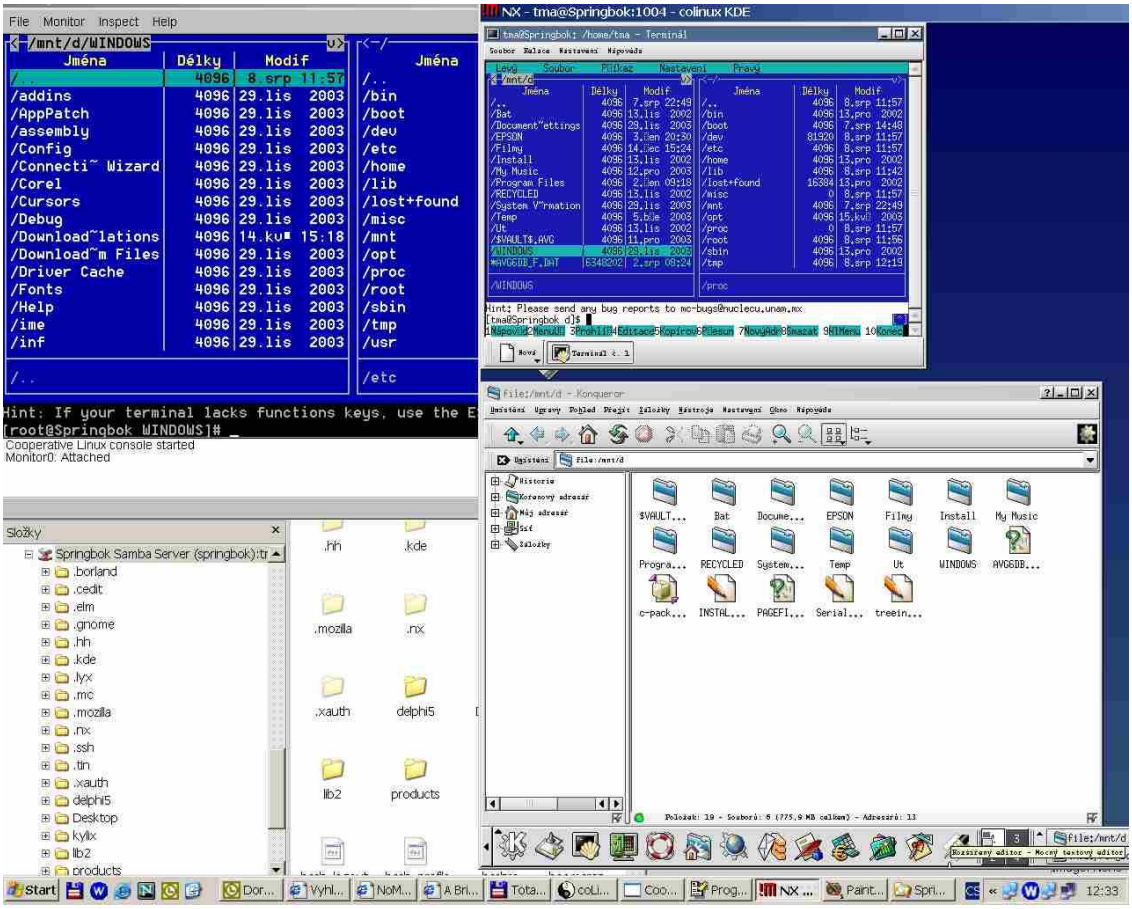
Freeware since v4.6.5, download from http://www.pexus.com/. Installation occupies approx 45MB of disk space.

NX Server

1. install "*NX Client for Linux*" on colinux (`rpm --nodeps` had to be used)
2. install "*NX Server Personal Edition*" at colinux (commercial ???)
3. install "*NX Client for Windows*" on Windows
4. add user `/usr/NX/bin/nxserver -useradd <linux_user>`
5. configure windows *nxclient*

Some info is at http://www.gnome.org/~markmc/a-look-at-nomachine-nx.html.


Cooperative Linux console
File Monitor Inspect Help
Jména Délky Modif
/.. 4096 7.srp 22:49
Hint: To look at the output of a command in the viewer,
[root@Springbok bin]# smb_retry: signal failed, error=-3
smb_lookup: find //Lib failed, error=-5
smb_lookup: find //Proj failed, error=-5
Cooperative Linux console started
Monitor0: Attached
NX - tma@Springbok:1004 - colinux KDE


File Monitor Inspect Help
/mnt/d/WINDOWS
Cooperative Linux console started
Monitor0: Attached
NX - tma@Springbok:1004 - colinux KDE

Common X-Server

1. `export DISPLAY=192.168.1.20:0`
2. Start your Windows X-Server
3. `xterm &`

3. Conclusion

coLinux is powerful GNU software that enables running Linux in Windows but some drawbacks exist:

1. coLinux is running as normal application and since get only a bit of machine time it's relatively slow
2. only 10Mb NIC is emulated
3. no Windows X-Server is included

I tested in the past *VMWare 2.03* but in opposite configuration – virtual Windows in X-Windows – and it worked perfectly. I expect that VMWare providing virtual Linux is comparable powerful and implements X-Windows server. But VMWare is commercial and is relatively expensive.