

---

# D68000 with uCLinux

## Testing platform

---



## Contents

<b>1. System overview</b>	<b>3</b>
1.1. Hardware setup	4
1.2. Memory map	5
1.3. Communication with host	6
<b>2. Target system overview</b>	<b>7</b>
2.1. D68000 microprocessor	7
2.2. On-chip peripherals	8
2.3. PCB board & off-chip components	8
<b>3. Host Software platform</b>	<b>8</b>
3.1. GCC overview	9
3.2. GDB overview	9
3.3. System Simulator	10
<b>4. uCLinux overview</b>	<b>11</b>
4.1. Configuration & compilation	11
4.2. General components	11
4.3. Uploading & running	12
<b>5. Example applications running with uCLinux</b>	<b>14</b>
5.1. Shell	14
5.2. HTTP Server	15
5.3. FTP client	16
5.4. Mounting NFS	17

## 1. SYSTEM OVERVIEW

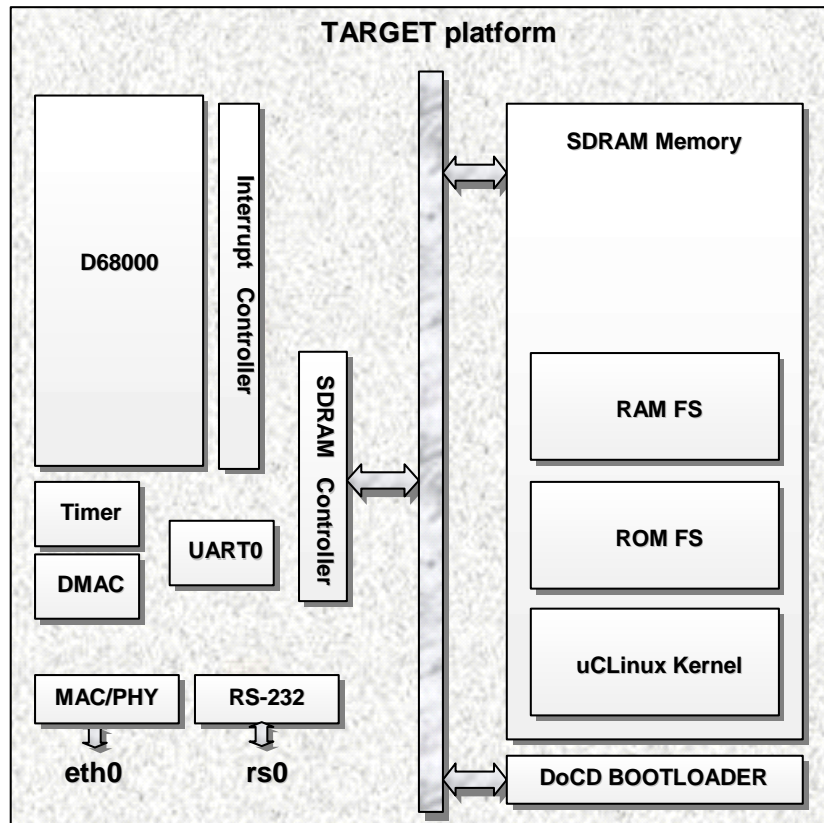
The main goal of this document is show that D68000 microprocessor core is fully functional and well validated IP Core. For that purpose complete testing system has been built basing on CYCLONE-2 2C20-6 FPGA board with some on-board memories and peripherals. The widely known Operating System for embedded solutions - uCLinux - has been used to run with D68000. Such combination of hardware and software created useful and flexible platform with D68000 IP Core as main processor.



The most of the applications and tools have been already available as GPL based software. As a source for all uCLinux based stuffs <http://www.uclinux.org/> site was used.

## 1.1. HARDWARE SETUP

The testing system is consisted of the following components presented in figure below.



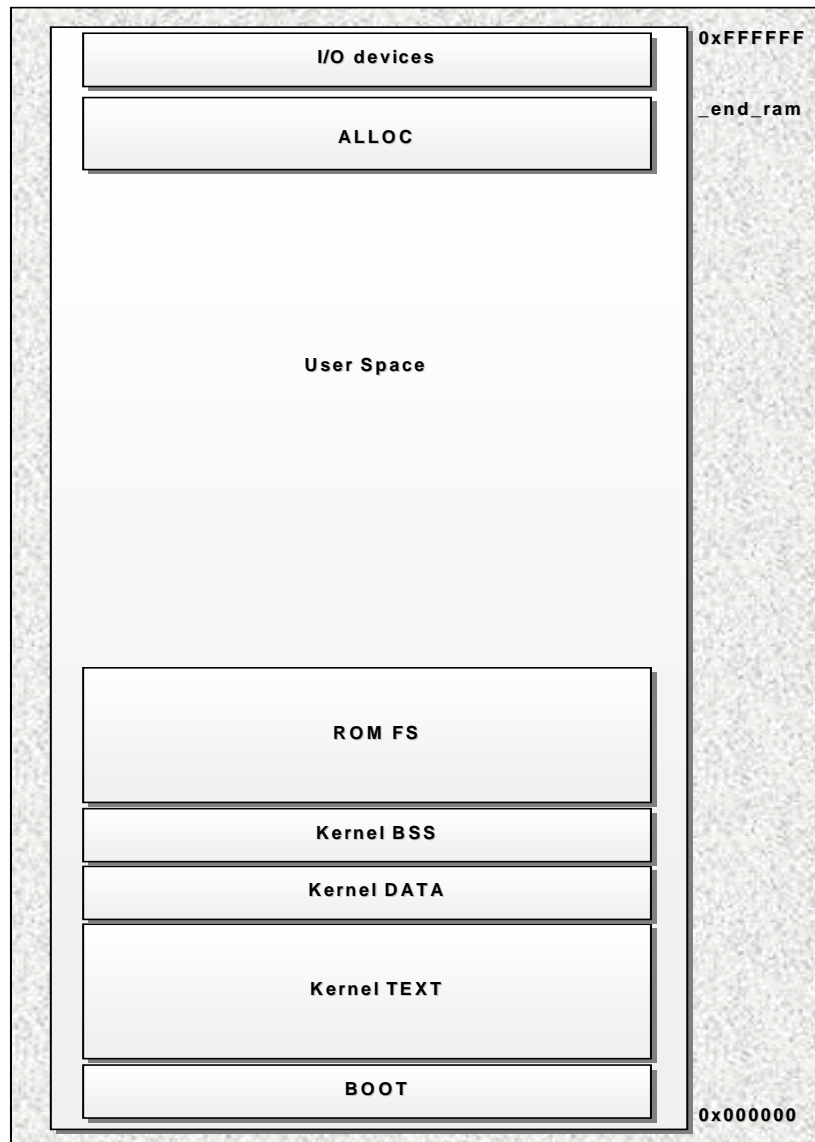
**TARGET System overview**

BDM-DoCD interface is used for uCLinux image upload from host. One serial RS-232 port is used as a main console for interactive communication between user and uCLinux. A 10/100 Mb Ethernet connector is used to plug twisted pairs LAN cable. It is controlled by DCD's DMAC IP Core.

## 1.2. MEMORY MAP

From programmer point of view whole memory space is divided on the following regions, starting from lowest (0x000000) to upper (0xFFFFF) address boundary:

- BOOT - this area is occupied by system boot-loader
- KERNEL - this area is occupied by uCLinux kernel image and divided onto
  - TEXT - Read Only executable code
  - DATA - Initialized data
  - BSS - not initialized data section
- ROMFS - here is placed Read Only root file system
- User Space - User space used for run-time application
- ALLOC - dynamically allocated space by run-time applications
- I/O devices - at the end of space I/O devices such as UARTs, MAC, Timer, Interrupt registers are located



**Memory mapping**

**1.3. COMMUNICATION WITH HOST**

Target board communicates with host system using one RS-232 serial port, and one 10/100 Mb Ethernet port. BDM-DoCD is used for boot code loading, and initial image of uCLinux upload from host platform. Ethernet connector and second serial port is used by running uCLinux operating system to communicate with user and outside world. For example main system console is provided through serial port, and Network File Systems are mapped by Ethernet connector, to extend available disk space for uCLinux.

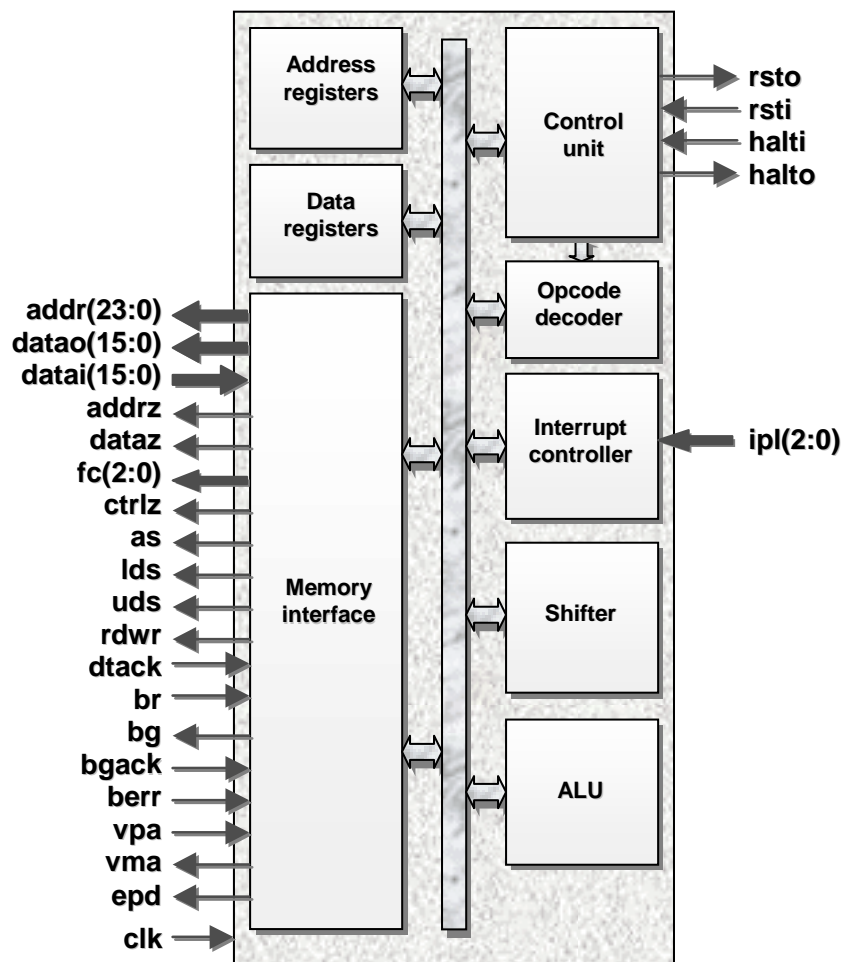
## 2. TARGET SYSTEM OVERVIEW

Target system is composed of FPGA development board with on-board memories and peripherals. It is configured to run D68000 microprocessor with added on-chip peripherals, creating System on Programmable Chip.

### 2.1. D68000 MICROPROCESSOR

This core is ideal while migration from standard PCB-based application using 68000 processor to modern System on-Chip is needed, and software running with 68000 must retain unmodified. The main features of D68000 microprocessor are:

- 100% binary compatible with 68000 industry standard
- full support of 68000 bus-cycle timings
- wide range of addressing modes
- improved architecture performance over original 68000
- broad range of standard software design tools including C/C++/ASM compilers, debuggers, and simulators
- target technology independence : any FPGA, as well as ASIC
- silicon proven



***D68000 block diagram***

## 2.2. ON-CHIP PERIPHERALS

The following peripherals have been added to the D68000 core, and compose fully working System on Chip.

- SDRAM controller - server large Synchronous Dynamic RAM memory
- Interrupt controller - manages interrupts from 32-bit timer, UART0 and DMAC controller
- 32-bit timer - serves precise in time interrupts needed by operating system
- one serial port - UART0 - establishes communication channel with external world using serial transmission
- MAC PHY interface - main LAN socket

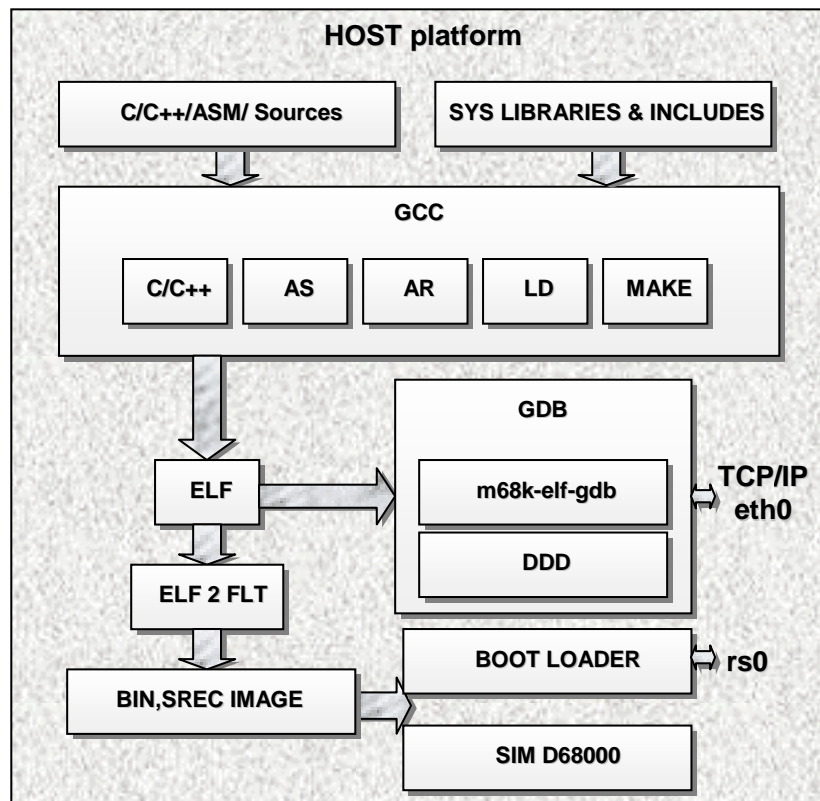
## 2.3. PCB BOARD & OFF-CHIP COMPONENTS

PCB board includes the following physical devices

- CYCLONE-2 2C20 -6 based FPGA
- 16 MB of SDRAM
- DMAC/ PHY, RS232, JTAG
- I/O Headers
- Push-buttons, LEDs
- Oscillator, voltage regulators

## 3. HOST SOFTWARE PLATFORM

Host software platform is built with GCC compiler, GDB debugger, and D68000 simulator. They are running with Windows OS. These set of tools is used to produce uCLinux system image including linux-2.4.20 kernel and applications running with kernel. It also assures instant simulation of compiled uCLinux system image, and checking how it works. The sources are available at <http://www.uclinux.org>.



**HOST system overview**

### 3.1. GCC OVERVIEW

GCC - GNU Compiler Collection is a high level C/C++ languages compiler. It provides complete set of tools allowing produce executable code for D68000 platform, libraries, and debug information needed by GDB. GCC main features:

- C/C++ languages
- complete set of libraries for these languages
- automation provided by make tool

### 3.2. GDB OVERVIEW

GDB - GNU Debugger is a C/C++/ASM source code debugging tool. An object code contains debugging information is produced by GCC. An executable code of application is loaded into target platform with D68000 and uCLinux. Host is running GDB, which communicates with target using TCP/IP. This communication method assures efficient and fast mechanism for debug data exchange. GDB main features:

- start a program, stop on specified conditions
- examine what has happened, when program has stopped
- change things in program, allowing experiments with correcting bugs

### 3.3. SYSTEM SIMULATOR

D68000 system simulator was built to provide instant method to verification of uCLinux system and its applications. It has built-in macro commands parser allowing automatic simulation of desired application. Its output results can be dumped to files and used in HDL simulator as reference data, as well as for debugging purposes.

## 4. uCLINUX OVERVIEW

uCLinux is a MMU-less derivative of Linux Operating System adopted for embedded solutions. It provides all of the Linux benefits including superior stability, Common Linux Kernel API, multitasking, full featured TCP/IP networking, Virtual File System, and reduces the amount of memory needed by its kernel and running applications. Basically uCLinux is not a RTOS, however preemptive kernel patches & low-latency patches are available. Actual version of uCLinux used with D68000 is based on Linux 2.4.x kernel.

### 4.1. CONFIGURATION & COMPILATION

uCLinux features are configurable by text/graphical user interface, the same way as standard Linux. In fact uCLinux exists within real Linux environments, and is adopted for certain architectures including D68000. Its kernel image in minimal configuration utilizes less than 512 kB, and utilities need about 400 kB.

### 4.2. GENERAL COMPONENTS

The following components of uCLinux has been selected and used in D68000 system.

- Linux kernel 2.4.x
  - *System V IPC - Inter Process Communication*
  - *Sysctl interface support*
  - *Kernel support shared FLAT binaries*
- Networking support
  - *Packet socket*
  - *Unix domain socket*
  - *TCP/IP networking*
- File systems
  - */proc FS*
  - *ROM FS*
  - *Second extended FS*
  - *Network File System*
- Core Applications
  - *ini & expand*
  - *Sash shell with system commands*
- Network Applications
  - *arp, ping, tcpdump*
  - *boa HTTP server, FTP cliet, nslookup*
  - *portmap, mount, umount, ifconfig, route, hostname*
- Miscellaneous Applications
  - *gdbserver, vi*
  - *date, ps, uptime*
  - *echo, find, du, df*

- *gzip, gunzip, tar*

### 4.3. UPLOADING & RUNNING

A small boot-loader has been written to upload system image and launch booting procedure of uCLinux. Boot-loader resides inside lower addresses of SRAM memory, handles UART events coming from host sending system image. After successful upload, kernel booting procedure starts. System booting process can be generalized to the following steps:

- initialize memory and hardware
- setup device drivers
- mount root file system
- execute /sbin/init
- start network service
- launch shell program

Booting uCLinux system is very fast and taking only few seconds. The reported boot messages are listed below:

```
Linux version 2.4.20-uc0 (root@back.comp.dcd.pl) (gcc version 2.95.3 20010315 (release)(ColdFire
patches - 20010318 from http://fiddes.net/coldfire/)(uCLinux XIP and shared lib patches from
http
://www.snapgear.com/)) #4 Wed Oct 22 19:46:40 CEST 2003

D68000 support (C) 2003 DCD

uCLinux/D68000
D68000 by DCD <www.digitalcoredesign.com>
Flat model support (C) 1998,1999 Kenneth Albanowski, D. Jeff Dionne
On node 0 totalpages: 1568
zone(0): 0 pages.
zone(1): 1568 pages.
zone(2): 0 pages.
Kernel command line: console=/dev/ttyS0 root=/dev/rom0
Calibrating delay loop... 2.63 BogoMIPS
Memory available: 4512k/6272k RAM, 0k/0k ROM (746k kernel code, 192k data)
Dentry cache hash table entries: 1024 (order: 1, 8192 bytes)
Inode cache hash table entries: 512 (order: 0, 4096 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 2048 (order: 1, 8192 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
D68000 serial driver version 0.25
ttyS0 at 0x00ffffffc (irq = 2) is a builtin D68000 UART
LAN91C111: smc_probe at 0x00FE0300 addr
SMSC LAN91C111 Driver (v2.01), (Linux Kernel 2.4 + Support for Odd Byte) 09/24/01 - by
Pramo
d Bhardwaj (pramod.bhardwaj@smsc.com)
eth0: SMC91C11xFD (rev:1) at 0xfe0300 IRQ:4 MEMSIZE:8192b NOWAIT:0 ADDR: 00:cf:49:52:01:c3
Blkmem copyright 1998,1999 D. Jeff Dionne
Blkmem copyright 1998 Kenneth Albanowski
Blkmem 1 disk images:
0: 11130C-1B470B [VIRTUAL 11130C-1B470B] (RO)
RAMDISK driver initialized: 16 RAM disks of 512K size 1024 blocksize
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 512 bind 512)
```

```

NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NET4: Linux IPX 0.47 for NET4.0
VFS: Mounted root (romfs filesystem) readonly.
Freeing unused kernel memory: 0k freed (0xf8000 - 0xf7000)
Shell invoked to run file: /etc/rc
Command: hostname uCFPGA.comp.dcd.pl.
Command: /bin/expand /etc/ramfs.img /dev/ram0
Command: mount -t proc proc /proc
Command:
Command: mount -t ext2 /dev/ram0 /var
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command:
Command: date 102219502003
Thu Oct 22 19:50:00 UTC 2003
Command:
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
Command: ifconfig eth0 10.10.10.111 netmask 255.255.255.0 broadcast 10.10.10.255
eth0: PHY=LAN83C183 (LAN91C111 Internal)
Command: route add 10.10.10.111 eth0
Command: route add default gw 10.10.10.100
Command: portmap &
[18]
Command: cat /etc/motd
Welcome to

```



For further information check:  
<http://www.uclinux.org/D68000> processor

```

Command: ifconfig
eth0      Link encap:Ethernet  HWaddr 00:CF:49:52:01:C3
          inet addr:10.10.10.111  Bcast:10.10.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 iB)  TX bytes:0 (0.0 iB)
          Interrupt:4 Base address:0x300

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 iB)  TX bytes:0 (0.0 iB)

Command: route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.10.10.111    0.0.0.0        255.255.255.255 UH    0      0      0 eth0
10.10.10.0      0.0.0.0        255.255.255.0  U     0      0      0 eth0
127.0.0.0       0.0.0.0        255.0.0.0      U     0      0      0 lo
0.0.0.0         10.10.10.100   0.0.0.0        UG    0      0      0 eth0
Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>

```

## 5. EXAMPLE APPLICATIONS RUNNING WITH UCLINUX

### 5.1. SHELL

SASH command shell is used for communication purpose between user and system. It allows execute all necessary system commands to manage files, directories, devices, run any applications. For example directories structure can be presented for user or new directory added, deleted, etc.

ROOT FS directory listing:

```

/> ls -la
drwxr-xr-x 1 root root 32 Jan 1 1970 .
drwxr-xr-x 1 root root 32 Jan 1 1970 ..
drwxr-xr-x 1 root root 32 Jan 1 1970 bin
drwxr-xr-x 1 root root 32 Jan 1 1970 dev
drwxr-xr-x 1 root root 32 Jan 1 1970 etc
drwxr-xr-x 1 root root 32 Jan 1 1970 home
drwxr-xr-x 1 root root 32 Jan 1 1970 lib
drwxr-xr-x 1 root root 32 Jan 1 1970 mnt
dr-xr-xr-x 19 root root 0 Nov 30 1999 proc
lrwxrwxrwx 1 root root 4 Jan 1 1970 sbin -> /bin
lrwxrwxrwx 1 root root 8 Jan 1 1970 tmp -> /var/tmp
drwxr-xr-x 1 root root 32 Jan 1 1970 usr
drwxr-xr-x 6 root root 1024 Nov 30 1999 var
/>
  
```

Create & delete an example file

```

/> cd /tmp
/var/tmp> ls -la
drwxr-xr-x 2 root root 1024 Oct 27 13:24 .
drwxr-xr-x 6 root root 1024 Nov 30 1999 ..
/var/tmp> touch test.file
/var/tmp> !2
drwxr-xr-x 2 root root 1024 Oct 27 13:25 .
drwxr-xr-x 6 root root 1024 Nov 30 1999 ..
-rw-r--r-- 1 root root 0 Oct 27 13:25 test.file
/var/tmp> rm test.file
/var/tmp> !3
drwxr-xr-x 2 root root 1024 Oct 27 13:26 .
drwxr-xr-x 6 root root 1024 Nov 30 1999 ..
/var/tmp>
  
```

## 5.2. HTTP SERVER

The BOA application is used as HTTP server. It is small quite good HTTP server. It runs example web page.

```

/> ps
  PID  PORT  STAT  SIZE  SHARED  %CPU  COMMAND
    1      S    90K   0K    0.0   0.0   init
    2      S     0K   0K    0.0   0.0   keventd
    3      S     0K   0K    0.0   0.0   ksoftirqd_CPU0
    4      S     0K   0K    0.0   0.0   kswapd
    5      S     0K   0K    0.0   0.0   bdflood
    6      S     0K   0K    0.0   0.0   kupdated
   18      S   110K   0K    0.0   0.0   portmap
   21     S0 S    90K   0K    0.0   0.0   /bin/sh
   22     S  90K   0K    0.0   0.0   /bin/boa
   60     S0 R    82K   0K    0.0   0.0   ps
/>
  
```



**WWW home page served by D68000-uClinux**

### 5.3. FTP CLIENT

FTP client is intended to establish communication through FTP protocol, with external host to exchange files. Example log from FTP session is included below:

```

/mnt> cd var
/mnt/var> ftp 10.10.10.80
Connected to 10.10.10.80.
220 FTP Server ready
Name (10.10.10.80:root): uclinux
331 Password required for uclinux.
Password:
230 User uclinux logged in.
ftp> cd D68000
250 CWD command successful.
ftp> ls test*
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--  1 root    root      1830 Feb 10  2003 test.c
-rw-r--r--  1 root    root     87432 Feb 10  2003 test.dmp
-rw-r--r--  1 root    root    159750 Oct 22  08:27 test.elf
-rw-r--r--  1 root    root     3913 Feb 10  2003 test.o
-rw-r--r--  1 root    root    12943 Feb 10  2003 test.odmp
-rwxr-xr-x  1 root    root    29246 Feb 10  2003 test.pic
226 Transfer complete.
ftp> bin
200 Type set to I.
ftp> hash
Hash mark printing on (1024 bytes/hash mark).
ftp> prompt
Interactive mode off.
ftp> get test.elf
local: test.elf remote: test.elf
200 PORT command successful
150 Opening BINARY mode data connection for test.elf (159750 bytes)
#####
#
#####
226 Transfer complete.
159750 bytes received in 2 secs (75 Kbytes/sec)
ftp> by
221 Goodbye.
/mnt/var> ls -la
drwxr-xr-x  2 root    root      4096 Dec 15  2003 .
drwxr-xr-x 11 root    root      4096 Nov 20  2003 ..
-rw-r--r--  1 root    root    159750 Oct 16  15:52 test.elf
/mnt/var>

```

## 5.4. MOUNTING NFS

uCLinux can mount Network File System exported by external Linux box. Thus allows uCLinux to extend its file system space to any amount that could be needed. NFS is of course visible as "normal" file system with ability to read, write and execute the files directly. It also allows in efficient way to develop an application on external host and then execute it within uCLinux without a need to upload or create another uCLinux image including the new application.

Mounting Network File System and listing its files:

```

/> mount 10.10.10.80:/tmp/www /mnt
/> ps
  PID  PORT  STAT  SIZE  SHARED  %CPU  COMMAND
    1      S    90K    OK    0.0    init
    2      S     0K    OK    0.0    keventd
    3      S     0K    OK    0.0    ksoftirqd_CPU0
    4      S     0K    OK    0.0    kswapd
    5      S     0K    OK    0.0    bdflush
    6      S     0K    OK    0.0    kupdated
   18      S   110K    OK    0.0    portmap
   21  S0 S    90K    OK    0.0    /bin/sh
   22      S    90K    OK    0.0    /bin/boa
   62      S     0K    OK    0.0    rpciod
   63      S     0K    OK    0.0    lockd
   64  S0 R    82K    OK    0.0    ps
/> ls -la /mnt
drwxr-xr-x  5 553      521      4096 Oct 21 17:17 .
drwxr-xr-x  1 root    root      32 Jan  1 1970 ..
drwxr-xr-x  2 root    root      4096 Oct 21 17:11 cgi-bin
drwxr-xr-x  2 553      521      4096 Apr 20 2000 images
-rw-r--r--  1 553      521      2439 Apr 20 2000 index.htm
-rw-r--r--  1 553      521      2439 Oct 16 16:02 index.html
-rw-r--r--  1 553      521         49 Apr 20 2000 robots.txt
drwxr-xr-x  2 553      521      4096 Apr 20 2000 stat
-rw-r--r--  1 root    root     159750 Oct 22 16:55 test.elf
/>

```

Changing a file attributes located on NFS

```

/> cd /mnt/
/mnt> ls -la test*
-rw-r--r--  1 root    root      159750 Oct 22 16:55 test.elf
/mnt>
/mnt> chmod 000 test.elf
/mnt> ls -la test*
-----  1 root    root      159750 Oct 22 16:55 test.elf
/mnt>

```