The Minix3 Notes

`http://rms46.vlsm.org/2/166.pdf`
Rahmat M. Samik-Ibrahim
vLSM.org, Pamulang 15417, Banten

Table of Contents

Login and Logout

You might need a user account (*name* and *password*) to login into the GNU/Linux system. There are too many ways on how to login and logout. Please contact your system administrator and ask on how to get a "shell prompt" or Xterminal. After having access, familiarize yourself with some *commands* and the "*vi*" editor. Make sure you know how to logout from the system. The logout command is either "**`logout`**" or "**`exit`**" or both.

Some Useful Command Lines (Shell)

1. **`man man`** -- an interface to the on-line reference manuals.
2. **`passwd`** -- change (own) password.
   a) **`passwd user`** -- change the password of user "user" (root only)
3. **`ls`** -- list directory contents.
   a) **`ls -al`** -- long list
   b) **`ls -alt`** -- long list, sorted by modification time
   c) **`ls -alS`** -- long list (GNU/Linux only), sorted by file size
4. **`cd directory`** -- change to directory
   a) **`cd`** -- change to default/home directory
   b) **`pwd`** -- show current directory
5. Basic File Utilities
   a) **`cp file1 file2`** -- copy file1 to file2
   b) **`rm file1`** -- remove (delete) file1
   c) **`mv file1 file2`** -- move (change) file1 to file2
   d) **`mkdir dir`** -- make directory dir
   e) **`rmdir dir`** -- remove directory dir
6. More File Utilities
   a) **`cat file`** -- read a file

b) **`more file`** -- read a file per screen

c) **`ln -s file sfile`** -- make a symbolic link from file to sfile

d) **`grep aworld file`** -- search string aword inside file

e) **`sort file`** -- sort a file

7. More Commands

a) **`top`** -- display systems task

b) **`find / -name minix3.iso -print`** -- find file minix.iso from the root (/)

c) **`chmod 755 file`** -- change file with access mode 755

d) **`chown user file`** -- change owner file to user

e) **`chgrp other file`** -- change group file to other

f) **`tar`** -- (tape) archive files

i. **`tar cf /tmp/tarfile.tar  directory/`** -- archive "directory/" into tarfile.tar

ii. **`tar tf /tmp/tarfile.tar`** -- list archive tarfile.tar

iii. **`tar xf /tmp/tarfile.tar`** -- extract archive tarfile.tar

Check List 1

You should be familiar with some basic commands like:

`(login/logout), man, passwd, ls, cd, pwd, cp, rm, mv, mkdir, rmdir, cat, more, ln, grep, sort, top, find, chmod, chown, chgrp, tar.`

VI (a greate editor)

1. **Basics**
   a) **`i`** -- insert (**`a`** -- append), enter the insert mode

   b) **`o`** -- open a line, enter the insert mode

   c) **`<esc>`** -- escape the insert mode to command mode

   d) **`q!`** -- quit

   e) **`wq!`** or **`ZZ`** -- write and quit

   f) **`h j k l`** -- move [left, down, up, right]

   g) **`r`** -- replace a character under cursor

   h) **`x`** -- delete a character under cursor

   i) **`u`** -- undo

2. **More advanced vi commands**

   a) **`d^`** -- delete from the beginning of line to the cursor

   b) **`d$`** -- delete from the cursor to the end of the line

   c) **`dd`** -- delete the whole line

   d) **`5dd`** -- delete 5 lines

   e) **`yy`** -- yank (copy) the line

   f) **`p`** -- put (paste) the line

   g) **`J`** -- joint current and next line

   h) **`:r file.txt`** -- read (insert) file.txt

   i) **`:w file.txt`** -- write the whole file into file.txt

   j) **`:1,8 w! file.txt`** -- write line 1 to 8 into file.txt

3. **Searching**

    a) **`/`** -- find forward

    b) **`?`** -- find backward

    c) **`1,$ s/^/xxx /`** -- substitute all line beginnings with "xxx"

    d) **`1,$ s/$/yyy/`** -- add "yyy" to all lines

Check List 2

You should be familiar with some basic commands like:

`(login/logout), man, passwd, ls, cd, pwd, cp, rm, mv, mkdir, rmdir, cat, more, ln, grep, sort, top, find, chmod, chown, chgrp.`

You should also be familiar with the vi editor.

Creating Your Own User Account

1. Boot again your MINIX3 system and login as “`root`”.

2. It's about time to have your own user account (eg. “`dullatip`” of group "999")

```
# mkdir /home/999/
# adduser dullatip other /home/999/dullatip
[processing a new user blah-blah-blah]
# passwd dullatip
Changing the shadow password of dullatip
New password:    [type-in-the-password]
Retype password: [re-type-it]
```

3. From now on, you should use your own user account whenever you see user “`dullatip`”.

Adding more Minix3 Packages

1. Let's add more packages into Minix3: **open-ssh, vim, rsync**
2. Login with user **root**.
3. The current package installer (Minix3 version 3.1.2a) searchers the internet for updates. This could be a problem if we are behind a firewall or if our network connection is slow. Therefore we should "fix" packman:
   `# elvis /usr/bin/packman`
   [Find all five '**`'http`**"s and replace them with a wrong protocol like "**`xttp`**"]
   Then, add package "open-ssh" (option [4]):

```
# packman
[blah-blah-blah Please choose:]
4. Let me select individual packages to install from CD or network.
Choice: [4]
OK, showing packages to install. [Blah-blah-blah RETURN]
No. Source Package Description
[Blah-blah-blah]
30  cdrom  openssh-4.3p2    openssh implementation of secure shell
[Blah-blah-blah]
Package to install? [RETURN for none] 30
Installing from /mnt/install/packages/openssh-4.3p2.tar.bz2 ..
Get source of openssh-4.3p2? (y/N) N
[Blah-blah-blah RETURN]
# shutdown
[Blah-blah-blah]
d0p0s0> boot d0p0
[Blah-blah-blah:  “3 Start Custom Minix 3” ]
Generating SSH2 RSA host key: Ok
Generating SSH2 DSA host key: Ok
[Blah-blah-blah]
Minix Release 3 Version 1.2a (console)
192.168.97.129 login:
```

4. Let's test the secure shell connection:

```
Minix Release 3 Version 1.2a (console)
192.168.97.129 login: dullatip
password: [type-in-the-password]
[blah-blah-blah message of the day]
Terminal type? (network) xterm
$ telnet localhost
Connecting to 127.0.0.1:23...
Connected
Minix  Release 3 Version 1.2a  (ttyp1)
192.168.97.129 login:
$ ssh localhost
[blah-blah-blah RSA fingerprint]
Are you sure you want to continue connecting (yes/no)? yes
[warning blah-blah-blah]
dullatip@localhost's password: [type-in-the-password]
[blah-blah-blah message of the day]
$ who
dullatip    console  Fri Sep 11 08:00
dullatip    ttyp0    Fri Sep 11 08:02 (localhost)
$
```

5. Let us try from the GNU/Linux host to the Minix3 system (Qemu):

```
$ telnet 192.168.97.129
$ ssh dullatip@192.168.97.129
```

6. Do not forget to install packages "vim" and "rsync" too.

Recompiling the Minix3 Kernel

1. After the login prompt, login as "`bin`" (same password then "root")
   ```
   Minix Release 3 Version 1.2a (console)
   192.168.97.129 login: bin
   password: [type-in-the-ROOT-password]
   ```
2. Let's make some modifications using elvis or vim or whatever editor. These modifications are just for the sake of showing a new recompiled kernel.
   a) Change directory to: "`cd /usr/src/`"
   b) Edit file: "`vim /usr/src/include/minix/config.h`"
      Change value "`OS_VERSION`" from "`1.2a`" to "`1.2aX`"
   c) Edit file: "`vim /usr/src/kernel/main.c`"
      Replace in "`kprintf()`" from "`MINIX`" to "`MeNeX: A modification of Minix`"
   d) Edit file: "`vim /usr/src/lib/posix/_uname.c`"
      Replace in "`strcpy()`" from "`Minix`" to "`MeNeX`"
3. Recompile the kernel with user "`bin`". It may take more than 10 minutes.
   ```
   $ make clean
   $ make world
   $ ls -al /boot/image
   total 906
   drwxr-xr-x 2 root  operator     192 May  3  2006 .
   drwxr-xr-x 4 root  operator     448 May  3  2006 ..
   -rw------- 1 root  operator  462336 May  3  2006 3.1.2a
   -rw------- 1 root  operator  462336 Sep 24 22:08 3.1.2aXr0
   $ shutdown
   [Blah-blah-blah]
   d0p0s0> boot d0p0
   [Blah-blah-blah:  "3 Start Custom Minix 3" ]
   MeNex Release 3 Version 1.2aX (console)
         192.168.97.129 login:
   ```

Check List 3

You should be familiar with some basic commands like:

`(login/logout), man, passwd, ls, cd, pwd, cp, rm, mv, mkdir, rmdir, cat, more, ln, grep, sort, top, find, chmod, chown, chgrp.`

You should also be familiar with the vi editor.
You should have a Minix3 system with additional:

a) username: <your-own-account>
b) packages: vim, open-ssh, rsync
c) a new kernel: "`MeNeX Release 3 Version 1.2aX (console)`" in `/boot/image/3.1.2aXr0`
d) allow to login from the GNU/Linux host with: `telnet` and `ssh`.

Backing Up Your Own Home Directory

1. (Minix) Using "`root`", clean the `/usr/archive/pub` directory :

```
# cd /usr/archive/pub

# rm -rf *
```

2. (Minix) Using your own user account. For example, user **dullatip**, attandence list **#06**, on 27 May 2008:

```
$ cd /home

$ tar cvf /usr/archive/pub/06-dullatip-080527.tar dullatip/

$ cd /usr/archive/pub/

$ bzip2 06-dullatip-080527.tar

$ ls
06-dullatip-080527.tar.bz2
```

3. (Linux Host) asuming Minix's IP is `192.168.97.129.`

```
$ cd ~/tmp

$ rsync -av rsync://192.168.97.129/pub/ ./
```

4. File "06-dullatip-`080527.tar.bz2`" is **now** in the "`~/tmp/`" directory.

Some Useful Functions

1. `accept()`: accept a connection on a socket
2. `atoi()`: convert a string to an integer
3. `bind()`: assigning a name to a socket
4. `connect()`: initiate a connection on a socket
5. `fgets()`: reads in characters from a stream
6. `gethostbyname()`: returns a structure of type host for the given host name
7. `listen()`: listen for connections on a socket
8. `memmove()`: copy from memory to memory
9. `memset()`: fill memory with bytes
10. `read()`: read from a file descriptor
11. `write()`: write from a file descriptor
12. `int socket(int domain, int type, int protocol):` a socket file descriptor to create an endpoint for communication.
    a) **domain**: AF_INET; internetwork: UDP, TCP, etc.
    b) **type**: SOCK_STREAM; provides sequenced, reliable, two-way, connection-based byte streams.
    c) **protocol**: 0; a single protocol
    d) Example: `sockfd = socket(AF_INET, SOCK_STREAM, 0);`

Some Examples

1. Try this following:

```
$ env <enter>

$ echo $USER <enter>
```

2. Compare above with this:

```
#include <stdio.h>
#include <stdlib.h>
main(void) {
   char *str;
   str=getenv("USER");
   printf("I am %s\n",str);
   str=getenv("EDITOR");
   printf("My editor is %s\n",str);
   exit(0);
}
```

Exercise 01: Process System Calls

1. You should have your own user account on your Minix3 system. Cross-check if "`rsync`" works so that you can transfer files **from/to** the Minix3 system.
2. Create directory "`ex01/`" inside your new home directory. Go inside that directory and create a new file "`report01.txt`". Use that file for reporting purposes. **Do not forget to write down your name**.
3. Study these following functions with "man" (manual) and write down a brief report:

   `getpid(), fflush(), fork(), waitpid()`
4. Write down this following program, "`multifork.c`". Compile the program by using "`cc -o multifork multifork.c`". Capture the output by running "`./multifork > multifork.txt`". Include it into the report.

```
/* multifork.c (c) 2005-2009 Rahmat M. Samik-Ibrahim, GPL-like */
/* *********** ************************************************ */
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define DISPLAY1 "PID INDUK** ** pid (%5.5d) ** ***********\n"
#define DISPLAY2 "val1(%5.5d) -- val2(%5.5d) -- val3(%5.5d)\n"
/************************************************* main ** */
main(void) {
   pid_t val1, val2, val3;
   printf(DISPLAY1, (int) getpid());
   fflush(stdout);
   val1 = fork();
   waitpid(-1,NULL,0);
   val2 = fork();
   waitpid(-1,NULL,0);
   val3 = fork();
   waitpid(-1,NULL,0);
   printf(DISPLAY2, (int) val1, (int) val2, (int) val3);
   exit (0);
}
```

5. Compare output "`multifork.txt`" with "`multifork1.txt`" where you delete functions "`fflush()`" dan "`exit()`".
6. Now, try to run this following "`isengfork.c`" file.

```
/* isengfork.c (c) 2007-2009 Rahmat M. Samik-Ibrahim, GPL-like */
/* *********** ************************************************/
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
main(void) {
   int ii=0;
   if (fork() == 0) ii++;
   waitpid(-1,NULL,0);
   if (fork() == 0) ii++;
   waitpid(-1,NULL,0);
   if (fork() == 0) ii++;
   waitpid(-1,NULL,0);
   printf ("Result = %3.3d \n",ii);
   exit(0);
}
```

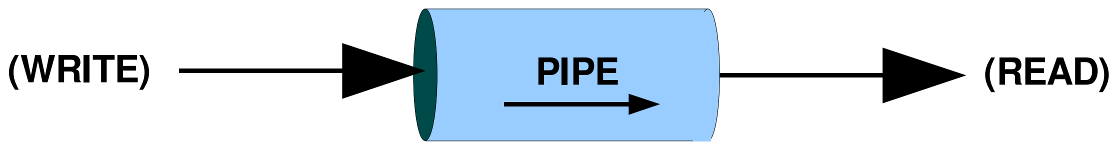
Exercise 02: Read/Write File

1. Create directory "`ex02/`" inside your home directory. Create a new file "`report02.txt`". Use that file for reporting purposes. **Do not forget to write down your name**.
2. Write down this following program, "`rw_file.c`". Study the related functions (`opendir(), readdir(), closedir(), time(), perror()`) with "man" and write down a brief report.
3. Compile it, run it, capture the output, and report it!

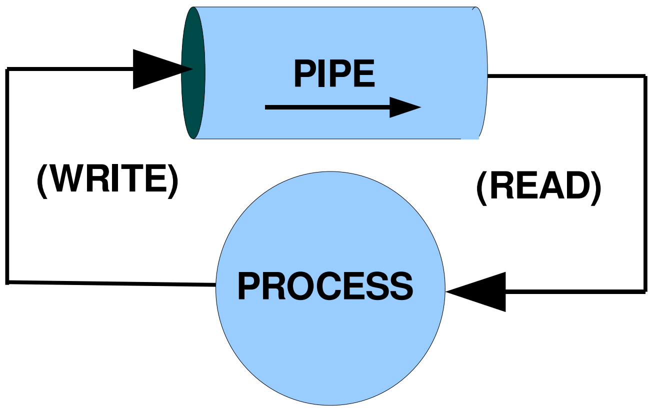
```
/* rw_file.c (c) 2007-2009 Rahmat M. Samik-Ibrahim, GPL-like */
/* ********* *************************************************/
#define  OLOOP 1000
#define  ILOOP 100
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <fcntl.h>
#include <dirent.h>
void rwfile (char *fname);
void dirfile(char *dname);
void error  (char *msg);
/* MAIN ============================== */
main(void) {
   printf("Listing current directory...\n");
   dirfile(".");
   printf("Testing read-write speed...\n");
   rwfile("normal.txt");
   exit(0);
}
/* DIRFILE ======================== */
void dirfile(char *dname) {
   DIR           *ddir;
   struct dirent *dp;
   printf("     ");
   ddir = opendir(dname);
   if (ddir != NULL) {
      while ((dp=readdir(ddir))!= NULL)
         printf("%s  ", dp->d_name);
      closedir(ddir);
   }
   printf("\n\n");
}
/* ERROR =========================== */
void error(char *msg){
   perror(msg);
   exit(0);
}
/* RWFILE =========================== */
void rwfile(char *fname) {
   time_t tt;
   int    fd, ii, jj;
   char   buf[] = "Achtung... Achtung... AAAA BBBB CCCC DDDD\n";
   time(&tt);
   for (ii=0;ii<OLOOP;ii++) {
      if ((fd=creat(fname,00644)) < 0 )
         error("RWFILE: can not create file\n");
      for (jj=0;jj<ILOOP;jj++)
         write(fd,buf,sizeof(buf)-1);
      close(fd);
   }
   tt=time(NULL)-tt;
   printf("Total time: %d seconds\n", (int) tt);
}
```

Exercise 03: PIPE

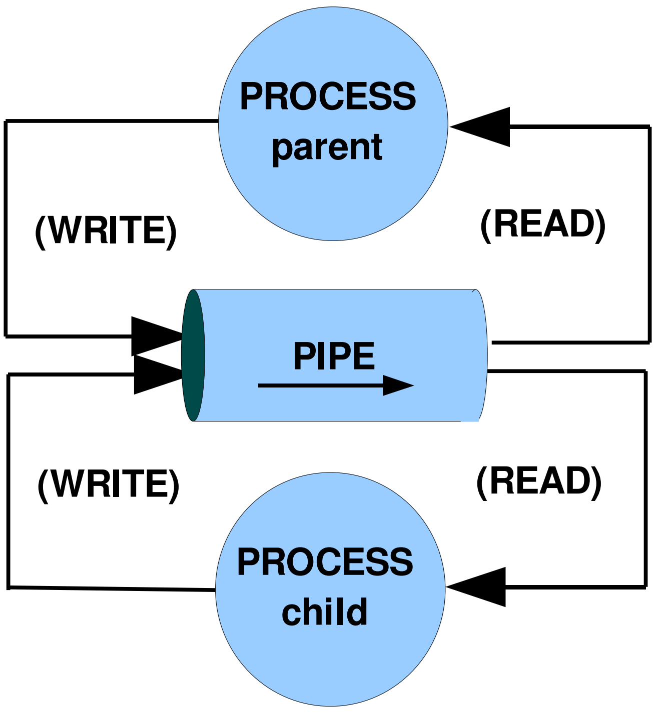
1. Create directory "`ex03/`" inside your home directory. Create a new file "`report03.txt`". Use that file for reporting purposes. **Do not forget to write down your name**. Study the related functions (`pipe(), fork(), close(), getpid(), write()`) with "man" (manual) and write down a brief report.
2. A pipe: you can write from one end, and read it from the other end. **WARNING:** Too many writes with no reads may cause the PIPE overflow and crash.



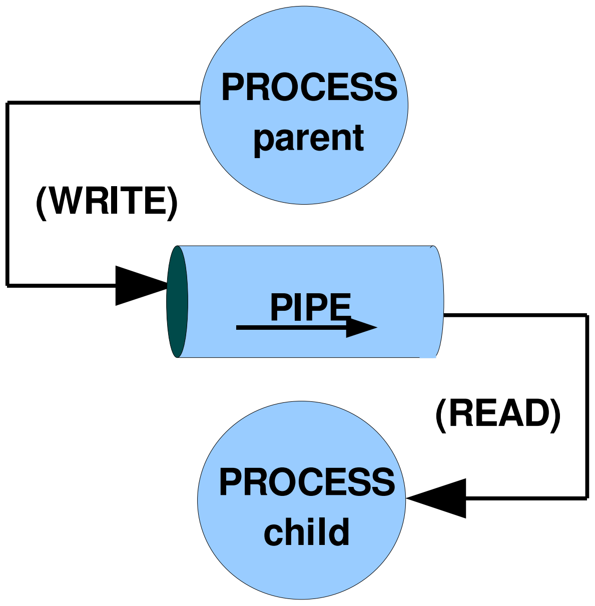
3. A process and a pipe (less fun).



4. A forked process and pipe (some fun). Whatever a process (parent or child) writes, can be read by both parent and child!



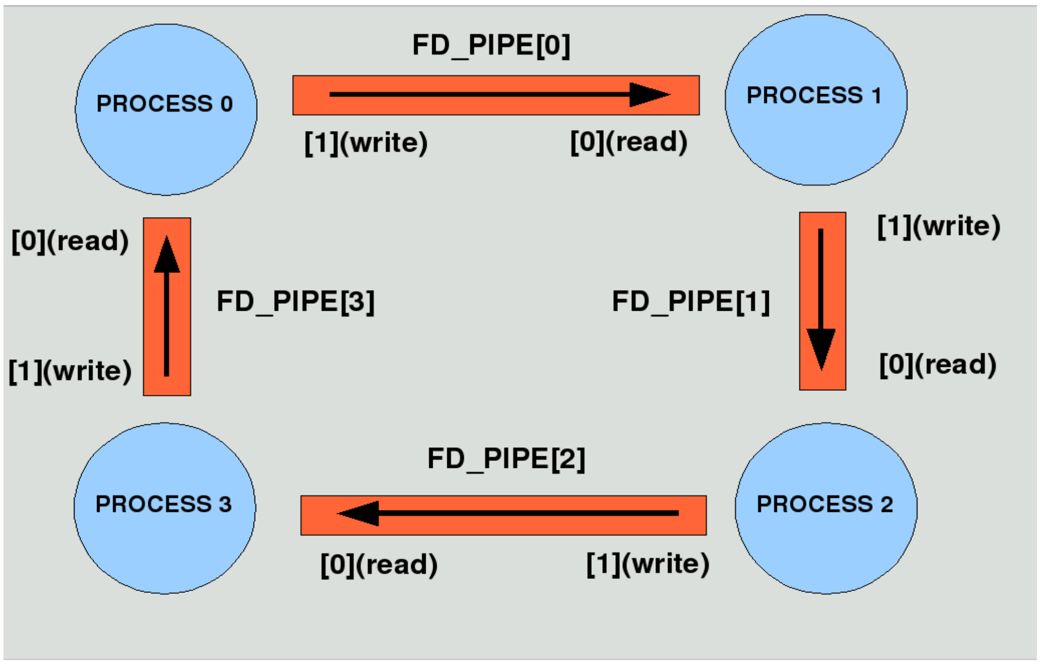
5. Same as above, but **disconnecting one write port and one read port.** Now, a parent can write to its child, or a child can write to its parent, or both!



6. Try this following program, "`forknpipe.c`".

```
/* forknpipe.c (c) 2007-2009 Rahmat M. Samik-Ibrahim, GPL-like */
/* ************ **************************************************/
#define  BUFSIZE 64
#define  WLOOP   5
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <string.h>

main(void) {
   char buffer[BUFSIZE];
   char message[]="Hello, what's up?\n";
   int  ii, pipe_fd[2];
   pipe(pipe_fd);
   if (fork() == 0) {
      /* child ****************************************/
      close(pipe_fd[0]);
      printf("I am PID[%d] (child).\n", (int) getpid());
      for (ii=0;ii<WLOOP;ii++)
         write(pipe_fd[1], message, sizeof(message)-1);
      close(pipe_fd[1]);
   } else {
      /* parent ***************************************/
      close(pipe_fd[1]);
      printf("I am PID[%d] (parent).\n",(int) getpid());
      memset(buffer, 0, sizeof(buffer));
      while ((ii=read(pipe_fd[0],
         buffer, BUFSIZE-1)) != 0) {
         printf("PARENT READ[%d]:\n%s\n", (int) ii, buffer);
         memset(buffer, 0,sizeof(buffer));
      }
      close(pipe_fd[0]);
   }
   exit(0);
}
```

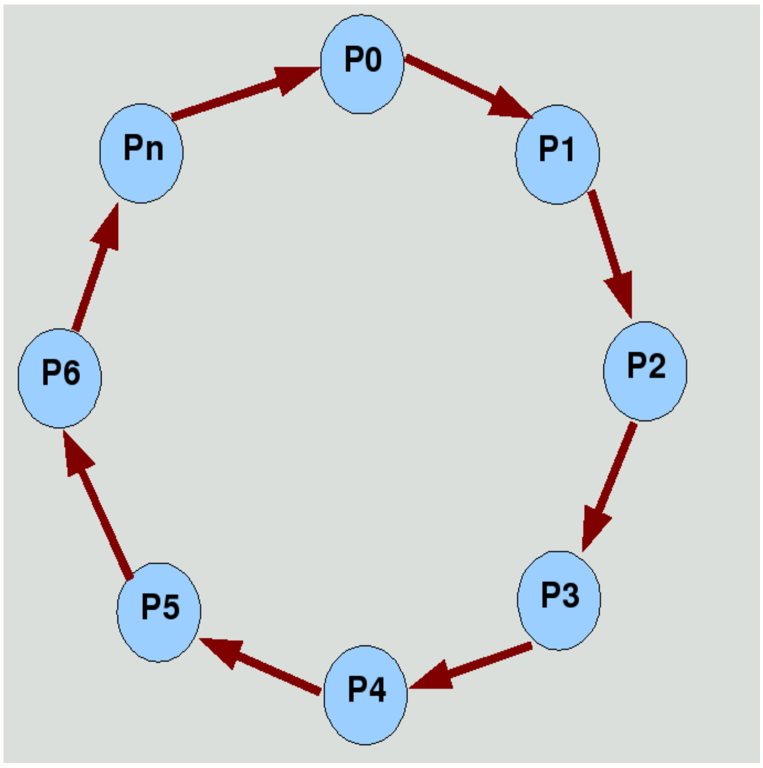
7. What if four processes?



8. Let's try this following "forknpipe2.c"

```
/* forknpipe2.c (c) 2007-2009 Rahmat M. Samik-Ibrahim, GPL-like */
/* ************ *************************************************/

#define  BUFSIZE 64

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <string.h>

main(void){
   char  buffer1[BUFSIZE],buffer2[BUFSIZE];
   int   p_this, p_prev, p_no1, p_no2;
   int   fd_pipe[4][2], ii, jj;
   pid_t mypid;

   memset(buffer1, 0, BUFSIZE);
   memset(buffer2, 0, BUFSIZE);

   for (ii=0;ii<4;ii++){
      pipe(fd_pipe[ii]);
   }

   ii = (fork() != 0 ) ? 0 : 2;
   jj = (fork() != 0 ) ? 0 : 1;

   p_this = ii + jj;
   close(fd_pipe[p_this][0]);

   p_prev = (p_this + 3) % 4;
   close(fd_pipe[p_prev][1]);
   p_no1  = (p_this + 1) % 4;
   close(fd_pipe[p_no1][0]);
   close(fd_pipe[p_no1][1]);

   p_no2  = (p_this + 2) % 4;
   close(fd_pipe[p_no2][0]);
   close(fd_pipe[p_no2][1]);

   mypid  = getpid();
   sprintf(buffer1,"   A message from PID[%d].\n", (int) mypid);
   write(fd_pipe[p_this][1], buffer1, BUFSIZE-1);
   close(fd_pipe[p_this][1]);

   while ((read(fd_pipe[p_prev][0], buffer2, BUFSIZE-1)) != 0) {
      waitpid(-1,NULL,0);
      printf("PID[%d] IS WAITING:\n%s\n", (int) mypid, buffer2);
   }
   close(fd_pipe[p_prev][0]);
   exit(0);
}
```

9. How about: P0 sends a message to P1, P1 forwards the message to P2, and so on. Last, Pn forward the message back to P0.



Exercise 04: Client and Server Programming

1. Create directory "`ex04/`" inside your home directory. Create a new file "`report04.txt`". Use that file for reporting purposes. **Do not forget to write down your name**.
2. Compile this following, "`server.c`", both on the Linux host and Minix system:

```
/* server.c Author: cut, pasted, and hacked until no error */
/* ******** ********************************************** */
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

void error(char *msg){
   perror(msg);
   exit(0);
}

int main(int argc, char *argv[]) {
   char   buffer[256];
   int    nn, sockfd, newsockfd;
   int    portno, clilen;
   struct sockaddr_in serv_addr;
   struct sockaddr_in  cli_addr;

   if (argc < 2) {
      fprintf(stderr, "ERROR, no port provided\n");
      exit(1);
   }
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0)
      error("ERROR opening socket");
   memset(&serv_addr, 0, sizeof(serv_addr));
   portno = atoi(argv[1]);
   serv_addr.sin_family      = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port        = htons(portno);
   if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr))< 0)
      error("ERROR on binding");
   listen(sockfd, 5);
   clilen = sizeof(cli_addr);
   newsockfd=accept(sockfd,(struct sockaddr *)&cli_addr,(socklen_t *)&clilen);
   if (newsockfd < 0)
      error("ERROR on accept");
   memset(buffer, 0, 256);
   nn = read(newsockfd,buffer,255);
   if (nn < 0)
      error("ERROR reading from socket");
   printf("Here is the message: %s\n",buffer);
   nn = write(newsockfd,"I got your message",18);
   if (nn < 0)
      error("ERROR writing to socket");
   return 0;
}
```

3. Compile this following, "`client.c`", both on the Linux host and Minix system.

```
/* client.c Author: cut, pasted, and hacked until no error */
/* ******** ************************************************/

#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

void error(char *msg) {
   perror(msg);
   exit(0);
}

int main(int argc, char *argv[]) {
   char   buffer[256];
   int    sockfd,     portno, nn;
   struct sockaddr_in serv_addr;
   struct hostent    *server;
   if (argc < 3) {
      fprintf(stderr, "usage %s hostname port\n", argv[0]);
      exit(0);
   }
   portno = atoi(argv[2]);
   sockfd = socket(AF_INET,SOCK_STREAM,0);
   if (sockfd < 0)
      error("ERROR opening socket");
   server =  (struct hostent *) gethostbyname(argv[1]);
   if (server == NULL) {
     fprintf(stderr, "ERROR, no such host\n");
     exit(0);
   }
   memset(&serv_addr,0,sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   memmove( &serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
   serv_addr.sin_port    = htons(portno);
   if(connect(sockfd, (const struct sockaddr *) &serv_addr, sizeof(serv_addr))<0)
       error("ERROR connecting");
   printf("Please enter the message: ");
   memset(buffer,   0, 256);
   fgets (buffer, 255, stdin);
   nn = write(sockfd,buffer,strlen(buffer));
   if (nn < 0)
      error("ERROR writing to socket");
   memset(buffer, 0, 256);
   nn = read(sockfd,buffer,255);
   if (nn < 0)
      error("ERROR reading from socket");
   printf("%s\n",buffer);
   return 0;
}
```

4. Try to send messages from the client to the server (how?)

a) Server: Minix -- Client: Minix

b) Server: Linux -- Client: Linux

c) Server: Linux -- Client: Minix

d) Server: Minix -- Client: Linux

5. Try to send a message to another host (your neighbor)

6. Try to pass a message from one host to the others.

user1 → user2 → user3 → ... → last_user.

Exercise 05: More Client/Server

1. Create directory "`ex05/`" inside your home directory. Create a new file "`report05.txt`". Use that file for reporting purposes. **Do not forget to write down your name**.
2. Compile this following, "`client_server.c`" and try it:

```
/* (c) 2007 Tadeus Prastowo, GPL-like
 *
 * This program serves as both a client and a server. Three modes of
 * operation are available:
 *  - initiating mode
 *  - bridging mode
 *  - terminating mode
 *
 * The following are how to run thisprogram for each mode:
 *  - Initiating mode:  client_server null ANOTHER_HOST ANOTHER_PORT
 *  - Bridging mode:    client_server CURRENT_PORT ANOTHER_HOST ANOTHER_PORT
 *  - Terminating mode: client_server CURRENT_PORT null null
 *
 * The program having the initiating mode _MUST_ run last after all other
 * instances of this program with other operational modes has been started.
 *
 * In initiating mode, this program just simply sends a hello message to
 * another instance of this program that operates either as a bridge or
 * as a terminator that this program points to as specified in
 * ANOTHER_HOST and ANOTHER_PORT. After that this program will quit
 * without printing out any message.
 *
 * In bridging mode, this program just simply waits for an incoming hello
 * message in CURRENT_PORT. Once it receives a hello message, it prints
 * out the message in a certain format. Next, this program forwards the
 * modified message to another instance of this program that acts either as
 * a bridge or as a terminator that this program points to as specified
 * in ANOTHER_HOST and ANOTHER_PORT. After that this program will quit.
 *
 * In terminating mode, this program just simply waits for an incoming hello
 * message in CURRENT_PORT. Once it receives a hello message, it prints out
 * the message in a certain format, and then quits.
 *

 * The following illustrates the idea above:

 * 192.168.10.18 (alvin)
 * $ ./client_server 8888 localhost 7777

 * 192.168.10.18 (user)$
 * $ ./client_server 7777 null null


 * 192.168.12.17 (eus)$
 * $ ./client_server null 192.168.10.18 8888


 * The print out will be:
 * 192.168.10.18 (alvin):
 *   From eus to alvin: Hello
 * 192.168.10.18 (user):
 *   From eus to alvin to user: Hello
 */
```

```
#define _MINIX

#include <stddef.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>

#define BUFFER_SIZE 256

void error (char *msg) {
   perror (msg);
   exit (0);
}

int main (int argc, char *argv []) {
   int sockfd, newsockfd, portno, clilen, count, nn;
   char buffer [BUFFER_SIZE], temp_buffer [BUFFER_SIZE], *colon_pos;
   struct sockaddr_in serv_addr, cli_addr;
   struct hostent *server;

   if (argc < 4) {
      fprintf (stderr,"\nUsage: %s this_port  next_sever next_server_port\n\n"
               "Start the chain with `this_port' = `null'\n\n"
               "Terminte the chain with `next_server' = `next_server_port'"
               " = `null'\n\n", argv [0]);
      exit (1);
   }
   if (strcmp (argv [1], "null") == 0) {
      portno = atoi   (argv [3]);
      sockfd = socket (AF_INET, SOCK_STREAM, 0);
      if (sockfd < 0) {
         error ("ERROR opening socket");
      }
      server = gethostbyname(argv[2]);
      if (server == NULL) {
         fprintf (stderr, "ERROR, no such host\n");
         exit (1);
      }
      memset (&serv_addr, 0, sizeof (serv_addr));
      serv_addr.sin_family = AF_INET;
      memcpy(&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
      serv_addr.sin_port = htons(portno);
      if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))< 0){
         error ("ERROR connecting");
      }
      /* Begin: action */
      memset (buffer, 0, BUFFER_SIZE);
      snprintf (buffer, BUFFER_SIZE, "From %s: Hello", getenv ("USER"));
      nn = write (sockfd, buffer, strlen (buffer));
      if (nn < 0) {
        error ("ERROR writing to socket");
      }
      /* End: action */
      exit (0);
   }
   sockfd = socket(AF_INET,SOCK_STREAM,0);
   if (sockfd < 0) {
      error ("ERROR opening socket");
   }
   memset(&serv_addr,0,sizeof(serv_addr));
   portno = atoi (argv [1]);
   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port = htons (portno);
   if (bind (sockfd,(struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
      error ("ERROR on binding");
   }
   listen (sockfd, 5);
   clilen    = sizeof (cli_addr);
   newsockfd = accept (sockfd, (struct sockaddr *) &cli_addr,
               (socklen_t *) &clilen);
   if (newsockfd < 0) {
      error ("ERROR on accept");
   }
   memset (buffer, 0, BUFFER_SIZE);
   nn = read(newsockfd,buffer,BUFFER_SIZE-1);
   if (nn < 0) {
      error ("ERROR reading from socket");
   }
   /* Modify buffer's message */
   colon_pos = strchr (buffer, ':');
   nn        = colon_pos - buffer;
   memset (temp_buffer, 0, BUFFER_SIZE);
   strncpy (temp_buffer, buffer, nn);
   memset (buffer, 0, BUFFER_SIZE);
   strncpy (buffer, temp_buffer, nn);
   snprintf (buffer + nn, BUFFER_SIZE-nn, " to %s: Hello", getenv ("USER"));
   /*End of modifying buffer's message*/
   if (strcmp (argv [2], "null") != 0 && strcmp (argv [3], "null") != 0) {
      portno = atoi (argv [3]);
      sockfd=socket(AF_INET,SOCK_STREAM,0);
      if (sockfd < 0) {
         error ("ERROR opening socket");
      }
      server = gethostbyname (argv [2]);
      if (server == NULL) {
         fprintf (stderr, "ERROR, no such host\n");
         exit (1);
      }
      serv_addr.sin_family = AF_INET;
      memcpy (&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
      serv_addr.sin_port = htons (portno);
      if (connect (sockfd,(struct sockaddr *)&serv_addr,sizeof (serv_addr))<0){
         error ("ERROR connecting");
      }
      /* Begin: action */
      printf ("%s\n", buffer);
      nn=write(sockfd,buffer,strlen(buffer));
      if (nn < 0) {
        error ("ERROR writing to socket");
      }
      /* End: action */
   } else {
      printf ("%s\n", buffer);
   }
   return 0;
}
```

Exercise 06: Performance

1. Create directory "`ex06/`" inside your home directory. Create a new file "`report06.txt`". Use that file for reporting purposes. **Do not forget to write down your name**.
2. First, write down text-file "`inputfile.txt`" with at least 1024 characters.
3. Second, write down a simple "`Makefile`". Take note: capital "**M**" in **M**akefile.

```
# Makefile
# (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
# ############################################################################

ALL:
        @echo " "
        @echo "make world -- make all"
        @echo "make clean -- clean the directory"
        @echo " "

world: myparent myfiles mypipes mysockets

clean:
        rm -f *.o
        rm -f myparent myfiles mypipes mysockets outputfile.txt
        @clear;
        @echo "================================================================"
        @ls
        @echo "================================================================"

myparent: myparent.o
        cc -o myparent myparent.o

myfiles: myfiles.o mycommon.o
        cc -o myfiles myfiles.o mycommon.o

mypipes: mypipes.o mycommon.o
        cc -o mypipes mypipes.o mycommon.o

mysockets: mysockets.o mycommon.o
        cc -o mysockets mysockets.o mycommon.o

myparent.o: myparent.c
        cc -c -o myparent.o myparent.c

myfiles.o: myfiles.c mycommon.h
        cc -c -o myfiles.o myfiles.c

mypipes.o: mypipes.c mycommon.h
        cc -c -o mypipes.o mypipes.c

mysockets.o: mysockets.c mycommon.h
        cc -c -o mysockets.o mysockets.c

mycommon.o: mycommon.c mycommon.h
        cc -c -o mycommon.o mycommon.c


# ############################################################################
```

4. Write down this following "`myparent.c`" file:

```
/* myparent.c
 * (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
 */

#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

char *children[]={"./myfiles", "./mypipes", "./mysockets"};

main(void)
{
   int ii;

   printf("MYPARENT:  start\n");
   for (ii=0;ii<3;ii++) {
      if (fork() == (pid_t) 0) {
         execve(children[ii],NULL,NULL);
      }
   }
   wait(NULL);
   wait(NULL);
   wait(NULL);
   printf("MYPARENT:  end\n");
   exit (0);
}
```

5. File "`myfiles.c`":

```
/* myfiles.c
 * (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
 */

#include "mycommon.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/times.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>

char   buf1[BFSIZ];

main(void){
   int        ii, fd;
   time_t     tt;
   clock_t    ctu, cts;
   struct tms tbuf;

   times(&tbuf);
   ctu = tbuf.tms_utime;
   cts = tbuf.tms_stime;
   time(&tt);
   printf("MYFILES:   start\n");
   if((fd=open(IFILE,O_RDONLY)) < 0)
      error("MYFILE: can not open file\n");
   memset(buf1, 0, BFSIZ);
   read(fd, buf1, BFSIZ-1);
   close(fd);
   for (ii=0; ii<MYLOOP;ii++) {
      if((fd=creat(OFILE,00644)) < 0)
         error("eMYFILE: can not create file\n");
      write(fd, buf1, BFSIZ);
      close(fd);
   }
   times(&tbuf);
   ctu = tbuf.tms_utime - ctu;
   cts = tbuf.tms_stime - cts;
   tt  = time(NULL)-tt;
   printf("MYFILES:total %d seconds (usr:%d sys:%d)\n", (int)tt, (int)ctu, (int)cts);
   exit(0);
}
```

6. File "`mypipes.c`":

```
/* mypipes.c
 * (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
 */

#include "mycommon.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/times.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>

char   buf1[BFSIZ];

main(void)
{
   int         ii, fd, pipe_fd[2];
   time_t      tt;
   clock_t     ctu, cts;
   struct tms  tbuf;

   times(&tbuf);
   ctu = tbuf.tms_utime;
   cts = tbuf.tms_stime;
   time(&tt);
   printf("MYPIPES:   start\n");
   pipe(pipe_fd);


   if (fork() == 0) {
      /* child */
      minidelay(MDELAY1);
      if((fd=open(IFILE,O_RDONLY)) < 0)
         error("MYFILE: can not open file\n");
      memset(buf1, 0, BFSIZ);
      read(fd, buf1, BFSIZ-1);
      close(fd);
      close(pipe_fd[0]);
      for (ii=0;ii<MYLOOP;ii++)
         write(pipe_fd[1], buf1, BFSIZ-1);
      close(pipe_fd[1]);
   } else {
      /* parent */
      close(pipe_fd[1]);
      while ((ii=read(pipe_fd[0], buf1, BFSIZ-1)) != 0) {
           memset(buf1, 0, BFSIZ);
      }
      close(pipe_fd[0]);
   }
   times(&tbuf);
   ctu = tbuf.tms_utime - ctu;
   cts = tbuf.tms_stime - cts;
   tt  = time(NULL)-tt;
   printf("MYPIPES:   total %d seconds (usr: %d sys: %d) -- PID[%d]\n",
                            (int) tt, (int) ctu, (int) cts, (int) getpid());
   exit(0);
}
```

7. File "`mysockets.c`":

```
/* mysockets.c
 * (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
 */
#include "mycommon.h"
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <sys/types.h>
#include <sys/times.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#include <netdb.h>

char   buf1[BFSIZ];

main(void)
{
   int                 ii, fd, pipe_fd[2];
   int                 sockfd, newsockfd, portno, clilen, count, nn;
   time_t              tt;
   clock_t             ctu, cts;
   struct tms          tbuf;
   struct hostent      *server;
   struct sockaddr_in serv_addr, cli_addr;

   times(&tbuf);
   ctu= tbuf.tms_utime;
   cts= tbuf.tms_stime;
   time(&tt);
   printf("MYSOCKETS: start\n");
```

```
if (fork() == 0) {
   /* child */
   if((fd=open(IFILE,O_RDONLY)) < 0)
      error("MYSOCKETS: can not open file\n");
   memset(buf1, 0, BFSIZ);
   read(fd, buf1, BFSIZ-1);
   close(fd);
   delay(DELAY1);

   sockfd = socket (AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0) {
      error ("MYSOCKETS opening socket");
   }

   server = gethostbyname (HOSTNAME);
   if (server == NULL) {
      fprintf (stderr, "MYSOCKETS: no such host\n");
      exit (1);
   }

   memset (&serv_addr, 0, sizeof (serv_addr));
   serv_addr.sin_family = AF_INET;
   memcpy (&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
   serv_addr.sin_port = htons (MYPORT);
   if (connect (sockfd, (struct sockaddr *) &serv_addr,
       sizeof (serv_addr)) < 0) {
           error ("MYSOCKETS connecting");
   }
   for (ii=0;ii<MYLOOP;ii++){
      if (write(sockfd, buf1, BFSIZ-1) < 0) {
         error ("MYSOCKETS writing to socket");
      }
   }
   close(sockfd);
} else {
   /* parent */
   sockfd = socket (AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0) {
      error ("MYSOCKETS opening socket");
   }
   memset (&serv_addr, 0, sizeof (serv_addr));
   serv_addr.sin_family      = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port        = htons (MYPORT);

   if (bind (sockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr))<0){
      error ("MYSOCKETS on binding");
   }
   listen (sockfd, 5);
   clilen = sizeof (cli_addr);
   newsockfd = accept (sockfd, (struct sockaddr *) &cli_addr,
      (socklen_t *) &clilen);
   if (newsockfd < 0) {
      error ("MYSOCKETS on accept");
   }
   memset (buf1, 0, BFSIZ);
   while ((ii=read (newsockfd, buf1, BFSIZ-1)) > 0)
      memset (buf1, 0, BFSIZ);
   if (ii < 0) {
     error ("MYSOCKETS reading from socket");
   }
   close(newsockfd);
}
```

```
   times(&tbuf);
   ctu = tbuf.tms_utime - ctu;
   cts = tbuf.tms_stime - cts;
   tt  = time(NULL)-tt;
   printf("MYSOCKETS: total %d seconds (usr: %d sys: %d) -- PID[%d]\n",
                             (int) tt, (int) ctu, (int) cts, (int) getpid());
   exit(0);
}
```

8. File "mycommon.c":

```
/* mycommon.c
 * (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
 */
#include "mycommon.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

/* MINIDELAY ========================= */
void minidelay(long duration)
{
   int ii;
   for (ii=0;ii<duration;ii++)
      ;
}

/* DELAY ============================= */
void delay(int duration)
{
   sleep(duration);
}

/* ERROR ============================= */
void error (char *msg)
{
   perror (msg);
   exit   (0);
}
```

9. File "`mycommon.h`":

```
/* mycommon.h
 * (c) 2007-2009 Rahmat M. Samik-Ibrahim -- rev 090222-02
 */

#define MYLOOP   10000
#define MYPORT   6666
#define HOSTNAME "localhost"
#define DELAY1   1
#define MDELAY1  1000
#define BFSIZ    1024
#define IFILE    "inputfile.txt"
#define OFILE    "outputfile.txt"

void minidelay(long duration);
void delay    (int  duration);
void error    (char *msg);
```

10. Run "`make world`'' in the current directory. Program "make" will search "Makefile" in the current directory.

Exercise 07: Disk Partitioning and Formating

1. Create directory "`ex07/`" inside your home directory. Create a new file "`report07.txt`". Use that file for reporting purposes. **Do not forget to write down your name**.
2. There exists two more vmplayer's "disks" with size 16 Mbytes and 2000 Mbytes with two additional devices: `/dev/c0d1` (16M) and `/dev/c0d3` (2000M). We are going to format `/dev/c0d1` directly with no partition and then devide `/dev/c0d3` into four (4) main partitions: `/dev/c0d3p0` (500M), `/dev/c0d3p1` (500M), `/dev/c0d3p2` (500M), `/dev/c0d3p3` (500M). Next, we are going to devide partition 3 into four (4) sub-partitions of about 128MBytes each: `/dev/c0d3p3s0`, `/dev/c0d3p3s1`, `/dev/c0d3p3s2`, and `/dev/c0d3p3s3`.
3. Most of the operations in Minix will need superuser privileges. Therefore in Minix, login as user `root`, and add two more directories: `/mnt1/` and `/mnt2/`. Compare `/mnt1/` after and before mount and report it in `report07.txt`. Formating with no partition is straight forward:

```
        #    mkdir /mnt1
        #    mkfs /dev/c0d1
        #    mount /dev/c0d1 /mnt1
        #    df
        #    cd /mnt1
        #    touch 1 2 3 4 5
        #    ls -al
        #    cd /
        #    umount /dev/c0d1
        #    ls -al /mnt1
```

4. (Minix) Next devide `/dev/c0d3` into four partitions with size about 500 Mbytes. Then devide `/dev/c0d3p3` into four subpatitions of about 128 Mbytes each.

```
        #    part /dev/c0d3
```

The display will be as following:

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device             Cyl Head Sec   Cyl Head Sec      Base      Size        Kb
    /dev/c0d3                        1015   64  63
                         0    0   0  1015   55  54         0   4096000   2048000
Num Sort   Type
 0    p0  00 None        0    0   0     0    0  -1         0         0         0
 1    p1  00 None        0    0   0     0    0  -1         0         0         0
 2    p2  00 None        0    0   0     0    0  -1         0         0         0
 3    p3  00 None        0    0   0     0    0  -1         0         0         0

Type '+' or '-' to change, 'r' to read, '?' for more help, 'q' to exit
```

The Minix partition type number is 81. Take note that the 'Base' of p1 is equal to 'Base + Size' of p0. And so on. Hit 'w' to write down/saving the partition table. After configuring, the partition should look like this following:

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device             Cyl Head Sec   Cyl Head Sec      Base      Size        Kb
    /dev/c0d3                         1015   64  63
                         0    0   0  1015   55  54         0   4096000   2048000
Num Sort   Type
 0*  p0  81 MINIX        0    1   0   253   63  62        63   1024065    512032
 1   p1  81 MINIX      254    0   0   507   63  62   1024128   1024128    512064
 2   p2  81 MINIX      508    0   0   761   63  62   2048256   1024128    512064
 3   p3  81 MINIX      762    0   0  1015   55  54   3072384   1023616    511808

Type '+' or '-' to change, 'r' to read, '?' for more help, 'q' to exit
```

Next, we are going to configure the sub-partition by hitting ">" at partition "p3":

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device             Cyl Head Sec   Cyl Head Sec      Base      Size        Kb
    /dev/c0d3                         1015   64  63
    /dev/c0d3:3        762    0   0  1015   55  54   3072384   1023616    511808
Num Sort   Type
 0   s0  00 None         0    0   0     0    0  -1         0         0         0
 1   s1  00 None         0    0   0     0    0  -1         0         0         0
 2   s2  00 None         0    0   0     0    0  -1         0         0         0
 3   s3  00 None         0    0   0     0    0  -1         0         0         0

Type '+' or '-' to change, 'r' to read, '?' for more help, 'q' to exit
```

Do not forget to hit "w" for saving the partition table. The result will be as following:

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device             Cyl Head Sec   Cyl Head Sec      Base      Size        Kb
    /dev/c0d3                         1015   64  63
    /dev/c0d3:3        762    0   0  1015   55  54   3072384   1023616    511808
Num Sort   Type
 0*  s0  81 MINIX      762    0   1   825   63  62   3072385    258047    129023
 1   s1  81 MINIX      826    0   0   889   63  62   3330432    258048    129024
 2   s2  81 MINIX      890    0   0   953   63  62   3588480    258048    129024
 3   s3  81 MINIX      954    0   0  1014   63  62   3846528    245952    122976

Type '+' or '-' to change, 'r' to read, '?' for more help, 'q' to exit
```

Now we have disk `/dev/c0d3` with these following partitions:

a) `/dev/c0d3p0` 512MB
b) `/dev/c0d3p1` 512MB
c) `/dev/c0d3p2` 512MB
d) `/dev/c0d3p3s0` 128MB
e) `/dev/c0d3p3s1` 128MB
f) `/dev/c0d3p3s2` 128MB
g) `/dev/c0d3p3s3` 128MB

5. Try to format `/dev/c0d3p3s3` and mount it to `/mnt2`.

```
#    mkfs /dev/c0d3p3s3
#    mount /dev/c0d3p3s3 /mnt2
#    df
#    cd /mnt2
#    touch 1 2 3 4 5
#    ls -al
```

6. Write down the report as usual. Cross check that your assignment has been copied properly.

FAP (Frequently Asked Problems)

1. Error Lists

   a) "`Could not open '/dev/kqemu' - QEMU acceleration layer not activated`" -- do not forget to add "`-no-kqemu`" in your linux start.

   b) "`qemu: syntax: -redir...`" -- fix the "redir" syntax

   c) "`qemu: could not open hard disk image 'disk.img'` " -- Disk image "disk.img" does not exists!

   d) "`qemu: could not open hard disk image 'minix.iso'` " -- CDROM image "minix.iso" does not exitsts!

   e) "`qemu: could not set up redirection`" -- there is another qemu running with the same "`-redir`" option.

2. Passwords

   a) "root" and "bin" share the same password. Do not forget our consensus.

   b) Do not forget, the password of "user" (our consensus).

   c) "shutdown" do not have a password.

3. The login prompt is not "`10.0.2.15 login:`" -- Minix does not recognize the ethernet emulation. Have you selected "4" (Realtek 8029) during installing Minix? Have you already set "qemu_pci=1"? What version of qemu do you use? There are some network problems with some qemu version 0.9.0.

4. Can not boot from "**`disk.img`**"

   In the minix startup, change "`-boot d`" to "`-boot c`"

5. Can not write to "**`disk.img`**"

   You should be the owner of "disk.img" and the mode should be “rw” (Read/Write).

6. RSYNC problems

Minix3 under Qemu

Note: This section is not maintained anymore.

1. **Check it out**: the Minix ISO file name may vary, assume it is "**`IDE-3.1.2a.iso`**".
2. Create a main disk:
```
$    qemu-img create disk.img 128M
```
3. Run the GNU/Linux version of qemu as following:
```
$     (nice -2 qemu -boot d -m 64 -hda disk.img       \
      -cdrom IDE-3.1.2a.iso       -localtime          \
      -no-kqemu   -net user       -net nic             \
      -redir tcp:5522::22         -redir tcp:5523::23 \
      -redir tcp:5873::873                            \
      -redir tcp:5524::5525)&
```
   Port numbers (5522, 5523, 5524, 5525, 5873) have to be unique if you run “qemu” on a multiuser system. The default Minix boot will be option [1]: "`Regular MINIX 3`"
4. Login with account "root"/no password and run "setup":
```
minix login: root
#  setup
```
5. Do these following steps:
   a) Step 1: select keyboard [us-std].
   b) Step 2: select ethernet chip [4. Realtek 8029].
   c) Step 3: select full installation [F].
   d) Step 4: create disk partition in automatic mode [ENTER].
      i. select default disk (/dev/c0d0 -- disk.img) [0]
      ii. select disk region [0]
      iii. confirm “yes”
   e) Step 5: Pass/omit this step (no prior home)
   f) Step 6: /home size [16]
   g) Step 7: block size [4]
   h) Step 8: check bad blocks in /dev/c0d0p0s0 (root), /dev/c0d0p0s1 (/home), and /dev/c0d0p0s2 (/usr).
   i) Step 9: copy the files.
6. Run:
```
#  shutdown
```
   Reboot the system and ignore the warning/error messages:
```
fd0> boot d0p0
```
   Login again with account "root" with no password and run "shutdown" again:
```
minix login: root
#  cp /etc/rc.daemons.dist /etc/rc.daemons
#  passwd
Changing the shadow password of root
New password:    [type-in-the-password]
Retype password: [re-type-it]
#  shutdown
```
   On the prompt:
```
d0p0s0>  qemu_pci=1
d0p0s0>  save
d0p0s0>  off
```
   Your minix is ready!

Minix3 Networking under Qemu

1. Boot your Minix from a qemu disk image on GNU/Linux ("**`3 Start Custom Minix 3`**"):

```
$  (nice -2 qemu -boot c -m 64 -hda disk.img       \
   -cdrom Minix-IDE-3.1.2a.iso  -localtime         \
   -no-kqemu   -net user        -net nic           \
   -redir tcp:5522::22         -redir tcp:5523::23 \
   -redir tcp:5873::873                            \
   -redir tcp:5524::5525)&
```

   Port numbers (5522, 5523, 5524, 5525, 5873) have to be unique if you run “qemu” on a multiuser system!

2. After the login prompt, login as "`root`":

```
Minix Release 3 Version 1.2a (console)
10.0.2.15 login: root
password: [type-in-the-password]
```

3. Wait for the prompt and add a new user:

```
# adduser user other /home/user
[processing a new user blah-blah-blah]
# passwd user
Changing the shadow password of user
New password:    [type-in-the-password]
Retype password: [re-type-it]
```

4. Testing the local telnet connection:

```
# telnet localhost
Connecting to 127.0.0.1:23...
Connected
Minix Release 3 Version 1.2a (ttyp0)
10.0.2.15 login: user
password: [type-in-the-password]
[blah-blah-blah message of the day]
Terminal type? (network) xterm
$ who
root    console  Fri Sep 11 08:00
user    ttyp0    Fri Sep 11 08:02 (localhost)
```

Installing Minix3 with a VMWare Player

1. The VMWare Player file set will be "`Generic.tar.gz`" or "`Generic.tar.bz2`" or "`Generic.zip`". There will be a file named "`Generic.vmx`" as following:

```
#!/usr/local/bin/vmware
.encoding          = "windows-1252"
displayName        = "Generic"
nvram              = "Generic.nvram"
extendedConfigFile = "Generic.vmxf"
memsize            = "96"
guestOS            = "dos"

floppy0.present    = "FALSE"
config.version     = "8"
virtualHW.version  = "6"
pciBridge0.present = "TRUE"
pciBridge0.pciSlotNumber = "17"

powerType.powerOff = "soft"
powerType.powerOn  = "hard"
powerType.suspend  = "hard"
powerType.reset    = "soft"

tools.upgrade.policy            = "useGlobal"
ft.secondary0.enabled           = "TRUE"
virtualHW.productCompatibility = "hosted"
vmotion.checkpointFBSize = "16777216"

uuid.location      = "56 4d 02 aa 9f 1e b8 f6-68 18 b4 a0 fb d6 87 85"
uuid.bios          = "56 4d 02 aa 9f 1e b8 f6-68 18 b4 a0 fb d6 87 85"
vc.uuid            = "52 06 b8 c0 71 0d dd 9e-8c 0d 7c 4d 88 de f5 20"

ide0:0.present      = "TRUE"
ide0:0.fileName     = "disk128M.vmdk"
ide0:0.writeThrough = "TRUE"
ide0:0.redo         = ""

ide0:1.present      = "TRUE"
ide0:1.fileName     = "disk16M.vmdk"
ide0:1.writeThrough = "TRUE"
ide0:1.redo         = ""

ide1:0.present      = "TRUE"
ide1:0.fileName     = "/extra/minix/minix3_1_2a_ide.iso"
ide1:0.deviceType   = "cdrom-image"
ide1:0.allowGuestConnectionControl = "FALSE"

ide1:1.present      = "TRUE"
ide1:1.fileName     = "disk2000M.vmdk"
ide1:1.writeThrough = "TRUE"
ide1:1.redo         = ""

ethernet0.present                = "TRUE"
ethernet0.allowGuestConnectionControl = "FALSE"
ethernet0.features               = "1"
ethernet0.wakeOnPcktRcv          = "FALSE"
ethernet0.networkName            = "NAT"
ethernet0.addressType            = "generated"
ethernet0.generatedAddress = "00:0c:29:d6:87:85"
ethernet0.pciSlotNumber          = "32"
ethernet0.generatedAddressOffset = "0"
```

Take note:

a) You might want to replace -- `displayName = "Generic"` -- with another name.
b) Memory size (**`memsize`**): 96 MBytes. To small memory size (less than 64 MB) causes some problem when running ssh.
c) IDE disk 0:0 (primary master): 128 MBytes; file-name: “`disk128M.vmdk`”.
d) IDE disk 0:1 (primary slave): 16 MBytes; file-name: “`disk16M.vmdk`”.
e) IDE disk 1:1 (secondary slave): 2 Gbytes; file-name: “`disk2000M.vmdk`”.
f) IDE CDROM 1:0 (secondary master): “`minix3.iso`”.
g) You still need a MINIX ISO Image. The image file name may vary, assume it is "`minix3.iso`". Replace “`ide1:0.fileName`” with a proper pathname of your **ISO image**.
h) Ethernet emulation: AMD LANCE
i) Network emulation: NAT

1. Run the VMWare Player -- either under GNU/Linux or MS/Windows -- and select “`Generic`” or whatever your replacement name is. Make sure, that it can boot Minix from the CDROM image.

2. There will be a warning, when running for the first time.
   When asked “`Did you move this virtual machine, or did you copy it?`”
   Answer with: “`I copied it`”

3. Wait until a Minix login prompt appears.

4. Login with account "root"/no password and run "setup":
```
minix login: root
#  setup
```

5. Do these following steps:
   a) Step 1: select keyboard [us-std].
   b) Step 2: select ethernet chip [6. AMD LANCE].
   c) Step 3: select full installation [F].
   d) Step 4: create disk partition in automatic mode [ENTER].
      i. select disk number [0] (`/dev/c0d0` -- 127 MB)
      ii. select disk region [0]
      iii. confirm “yes”
   e) Step 5: Pass/omit this step
   f) Step 6: /home size [16]
   g) Step 7: block size [4]
   h) Step 8: check bad blocks in `/dev/c0d0p0s0` (root), `/dev/c0d0p0s1` (/home), and `/dev/c0d0p0s2` (/usr).
   i) Step 9: copy the files.

6. Run:
```
#  shutdown
```
   Reboot the system and ignore the warning/error messages:
```
fd0> boot d0p0
```
   Login again with account "root" with no password and run "shutdown" again:
```
minix login: root
#  cp /etc/rc.daemons.dist /etc/rc.daemons
#  passwd
Changing the shadow password of root
New password:    [type-in-the-password]
Retype password: [re-type-it]
#  shutdown
```

```
d0p0s0> off
```

Your minix is ready!
PS: Do not forget the root password!

Shutdown

1. Add a special user, "shutdown":

```
# adduser shutdown operator /home/shutdown
[processing a new user blah-blah-blah]
```

2. Edit the profile of user "shutdown":

```
# elvis /home/shutdown/.profile
```

   a) add to the end of the .profile: "`/usr/bin/shutdown`"

   b) save the "`.profile`"

3. Test login with user "shutdown". The system should shut down. (**Before shutdown: make sure that no one is login into the system!**).

Rsync on Minix3

1. Using user "**root**", create directory "**`/usr/archive/`**" with mode=`755`; owner=**`dullatip`**; group=**`other`**. Replace "**`dullatip`**" with your own user-name.
```
# mkdir /usr/archive
# cd /usr/archive
# mkdir etc log pub
# chmod -R 755 .
# chown -R dullatip .
# chgrp -R other .
# chmod 777 pub
```
2. Create file **`/etc/rsyncd.conf`**:
```
motd file = /usr/archive/etc/motd.txt
log file  = /usr/archive/log/log.txt
[pub]
   comment   = This is MINIX PUB
   path      = /usr/archive/pub
   read only = yes
   list      = yes
   uid       = nobody
   gid       = nogroup
```
3. Create a startup file **`/usr/local/etc/rc.d/startrsync.sh`**
```
#! /bin/sh
/usr/local/bin/rsync --daemon
exit 0
```
4. Set the file above with mode 755
```
# chmod 755 /usr/local/etc/rc.d/startrsync.sh
```
5. Reboot the minix system, login with user "**user**" and watch
```
$ tail -f /usr/archive/log/log.txt
```
6. Create file **`/usr/archive/etc/motd.txt`**:
```
=========================================
This is MOTD of the MINIX Rsync Archive!
[YOUR INITIAL HERE]
=========================================
```
7. Fill **`/usr/archive/pub/`** with dummy files
```
$ cd /usr/archive/pub
$ mkdir test1 test2 test3
$ touch file1 file2 file3
$ ls -al
```
8. Test from Minix (user "**dullatip**"):
```
$ rsync rsync://localhost/
$ rsync rsync://localhost/pub/
```
9. Test from Linux Host -- angon -- (user "**dullatip**"). Assume your Minix IP is **`192.168.97.129`**.
```
$ rsync rsync://192.168.97.129/
$ rsync rsync://192.168.97.129/pub/
```
(Now you can copy files from **Minix** to **Linux**!)
```
$ cd ~ ; mkdir tmp ; cd tmp/
$ rsync -av rsync://192.168.97.129/pub/ pub/
$ cd pub ; ls
```

Rsync on GNU/Linux

1. Check with your local administrator if "rsync" is provided on the GNU/Linux system. If not, you

need to set up rsync with a private port (not 873).

2. Let's prepare the directories and files in **`/home/minix/archive`**. Replace **`/home/minix/`** with whatever available directory. Ask your local administrator/lab people.

```
$ cd /home/minix
$ mkdir archive
$ cd archive
$ mkdir etc log pub
$ cd pub
$ mkdir ltest1 ltest2 ltest3
$ touch lin1 lin2 lin3
$ ls -al
$
```

3. Create file **`/home/minix/archive/etc/rsyncd.conf`**

```
motd file = /home/minix/archive/etc/motd.txt
log file  = /home/minix/archive/log/log.txt
[pub]
   comment    = This is MINIX PUB
   path       = /home/minix/archive/pub
   use chroot = no
   read only  = yes
   list       = yes
   uid        = nobody
   gid        = nogroup
```

4. Create file **`/home/minix/archive/etc/motd.txt`**

```
========================================
This is MOTD of the LINUX Rsync Archive!
[YOUR INITIAL HERE]
========================================
```

5. Create script **`rsync-start.sh`** with mode 755:

```
#! /bin/sh
CONFILE="/home/minix/archive/etc/rsyncd.conf"
ROPTION="--daemon --port 5555''
rsync $ROPTION --config $CONFILE
exit0
```

6. Test it from Linux Host

```
$ rsync rsync://localhost:5555/pub/
```

7. Test it from Minix

```
$ rsync rsync://[LINUX.IP.ADDRESS]:5555/pub/
```

Note: there should be only one rsync home on a host with a unique port 5555.

References and URLs

This Minix3 Notes was cut and pasted from here, there, and everywhere. See also:

1. The Minix3 Webpage, `http://www.minix3.org/` , last visited on 9 September 2009.