Stagisti
Andrea Corà
Roberto Rinaldi

Tutore
Giuseppe Fortugno

OUR SUMMER STAGE TO INFN

It is the first time that high school students take part in the INFN summer stages; for this reason the lessons and the kinds of activities done are quite different by the ones of the previous years, when the stage-boys came from the university and from a deeper preparation about the subjects of their stages.

After a short (but rich) welcoming speech by the INFN director, every stager was assigned to his tutor.
Our stage theme was informatics.

During the first days we visited the Kloe's calculus center and we could see the several machines that are working together to receive and elaborate huge quantities of data from the homonymous experiment.
We could see Dafne and his functions, and other particular areas of the LNF.

During the rest of the stage time the LNF made the master room available, allowing us the use of several pcs.

The main aim of this 20 days experience of stage was to learn what a Unix operating system is and obviously how it works. The pcs dispose of a Linux operating system and of a connection to the other machines of the LNF through a LAN. We could record us as users with a personal name (stages2, 3) and a password to accede into the servers of the Kloe's calculus center.

The first days the lessons dealt with the general way of working of a computer and its main parts: starting to tell about the different parts of a processor and their functions (as the ALU unit, the registers..), we told about the hard disks, how they are done physically and how they are logically organized (how data are recorded on them and how they are read) through different kinds of file systems.

We told about the net, when and how it was born (through different Unix machines working in parallel way under the TCP/IP standards); we told about the LAN and the WAN, about other kinds of standards like the IGRP which allows quadrilateral linking among a certain number of machines.

Later our tutor told us about the several components of a Unix (the central nucleus-the kernel- and the exterior stratifications-the shells), and the concept the Unix is based on: the open source, also comparing it to other kinds of operating systems as Windows.

We could learn the main commands of a Unix, to list, open, read, show, modify files, to connect to another server, to save, move data or delete files.

We learn to use an editor, the VI, with its particular commands to create, modify or clear files or programs.

As the 90% of a Unix operating system is programmed in c language, we could see its main functions and code tags writing some programs (placed into appendix A).

We could realize that the c programming language is very "powerful", useful and used as I said before; we could notice the several c sides similar to the other programming language we knew: pascal, but we could realize that all the high level programming language are "different ways to say the same things": In fact the same algorithm can be written in c but also in pascal and in many other languages.

In a second time our lessons concerned with the installation of a Linux operating system on one of the computers of the master room; (this operation is described in the appendix B).

During the stage we could use several books to make our knowledge deeper; (these texts are declared into the appendix c).

Through this stage we learnt many things and above all we develop an interest with informatics. I think this stage shown us the wide and complicate world in which one day we'll work.

APPENDIX A

We wrote several programs in c language: the first of them is based on an old Eratosthenes's algorithm: it writes all the prime numbers under a certain numeric limit imposed by who makes the program run; here there're two different programs which do the same thing:

```
#include <stdio.h>
main ()
        {
        int a, n, i, count;
        scanf ("%d", &a);       /*takes data from the user and inserts it into the address
                                of a variable a*/
        printf ("   1\n");
        for (n = 2; n <= a; n++)        /*analyzes every number between 2 and the
                                        number set from the user*/
                {
                count = 0;
                for (i = 2; i < n; i++)   /*controls if n is a prime number*/
                        {
                        if (n % i != 0)
                                count++;
                        }
                if (count == n - 2)       /* if n is a prime number, writes it*/
                        printf ("%4d\n", n);
                }
        }
```

```
#include <stdio.h>
main()
        {
        int n, a, i, pippo;
        scanf ("%d", &a);
        printf ("   1\n");
        printf ("   2\n");
        for (n = 3; n <= a; ++n)
                {
                pippo = 0;
                for ( i = 2; i < n; ++i)
                        {
                        if (n % i == 0)
                                ++pippo;
                        }
                if (pippo == 0)
                        printf ("%4d\n", n);
                }
```

These are two programs with the same aim: they write how many prime numbers exist under a certain numeric limit imposed by who makes the program run:

```
#include <stdio.h>
main()
	{
	int a, n, i, count, prim;
	prim = 0;
	scanf ("%d", &a);
	for (n = 2; n <= a; n++)
		{
		count = 0;
		for(i = 2; i < n; i++)
			{
			if (n % i != 0)
			count++;
			}
		if (count == n - 2)
			prim++;		/*counts the prime numbers*/
		}
	printf ("%d\n", prim + 1);
	}
```

```
#include<stdio.h>
main()
	{
	int n, a, i, pippo, number;
	scanf ("%d", &a);
	number = 2;
	for (n = 3; n <= a; ++n)
		{
		pippo = 0;
		for (i = 2; i < n; ++i)
			{
			if (n % i == 0)
				++pippo;
			}
		if (pippo == 0)
			++number;
		}
	if (a == 1)
		printf ("   1\n");
	else
		printf ("%d", number);
	}
```

This is the longer and harder-to-write program we wrote: this program allows to order all files composed by several rows, every row composed by a string and a number divided by one space; the program allows to order the files in a numeric order or in a alphabetic order depending on a command the user gives to the program: once it's running the user can decide which kind of order he prefers writing a number if desires a numeric order, or writing a character if he prefers an alphabetic order:

```
#include <stdio.h>
#include <string.h>
void ordname (int i, char *a[2000], int b[2000]);
void ordnumber (int i, char *a[2000], int b[2000]);   /*initializes two function with three arguments
                                                       and that don't return anything*/
main ()
        {
        FILE *fp;                                     /*initializes a pointer to a file*/
        char *a[2000];                                /*initializes a vector of pointers to characters*/
        char buf1[2000][100];
        char x;
        int y, i;
        int b[2000];
        i = 0;
        fp = fopen ("file.c", "r");                   /*opens a file*/
        while (fscanf (fp, "%s %d", buf1[i], &b[i]) == 2)     /*inserts file's data into two vectors*/
                {
                a[i] = buf1[i];
                i++;
                }
        if (scanf ("%d", &y))
                ordnumber (i, a, b);                  /*if the user inserts a number, the program calls the
                                                       function ordnumber*/
        else if (scanf ("%s", &x))
                ordname (i, a, b);                    /*if the user inserts another character, the program
                                                       calls the function ordname*/
        }

void ordname (int i, char *a[2000], int b[2000])
        {
        int count, rc, m;
        char *n;
        count = 1;
        while (count != 0)
                {
                count = 0;
                for (rc = 0; rc < i; rc++)
                        {
                        if (strcmp (a[rc], a[rc + 1]) > 0)
                                {
                                n = a[rc];
```

```
                                m = b[rc];
                                a[rc] = a[rc + 1];
                                b[rc] = b[rc + 1];
                                a[rc + 1] = n;
                                b[rc + 1] = m;
                                count++;        /*if the first string is major than the second, the
                                                program exchanges them*/
                                }
                        }
                }
        for (rc = 1; rc < i + 1; rc++)
                printf ("%-20s %10d\n", a[rc], b[rc]);
        }

void ordnumber (int i, char *a[2000], int b[2000])
        {
        int count, rc, m;
        char *n;
        count = 1;
        while (count != 0)
                {
                count = 0;
                for (rc = 0; rc < i; rc++)
                        {
                        if (b[rc] > b[rc + 1])
                                {
                                n = a[rc];
                                m = b[rc];
                                a[rc] = a[rc + 1];
                                b[rc] = b[rc + 1];
                                a[rc + 1] = n;
                                b[rc + 1] = m;
                                count++;        /*if the first number is major than the second, the
                                                program exchanges them*/
                                }
                        }
                }
        for (rc = 1; rc < i + 1; rc++)
                printf ("%-20s %10d\n", a[rc], b[rc]);
        }
```

Back

APPENDIX B

The installation of the Linux has been done with a Redhat version 7.2, and it can be divided into several parts:

- **Declaration of the language during the installation itself;**
- **Declaration of the main hardware of the machine (mouse, keyboard and monitor);**
- **Declaration of the installation way (we used fdisk);**
- **Partition of the hard disk (choose how many partitions have to be created and how wide they have to be); (*)**
- **Declaration of the directories containing the partitions of the hard disk and declaration of the file system used for the partitions; (**)**
- **Declaration of the loader (grub or lilo-we used lilo);**
- **Declaration of the IP address, the netmask, the Default Route, the DNS (we left the standard DHCP);**
- **Declaration of the security rate of the Firewall;**
- **Declaration of the language and the time you prefer for your Linux;**
- **Declaration of the names and the passwords of the users;**
- **Declaration of the tools and packages you want on your Linux (kde or gnome graphic interfaces, games, and many others-these tools'll occupy a lot of memory);**
- **Installation of the Linux (it'll take several minutes);**
- **Declaration of the way to start the system (graphical interface or text interface);**

(*) There are several kinds of partitions. The primary partition, the extended partition and the logical partition: the primary partitions can be at last four but through the creation of an extended partition we can create more of four logical partitions into the extended one (the extended partition can't contain any primary partition). Every partition can contain a different operating system depending on the kind of file system used in that partition. A certain part of memory can be allocated as RAM through the swap (it isn't really RAM but it'll be used like this by Unix).
(**) We did four logical partitions into an extended one; two of them are dedicated to a Linux, one to Windows 2000, one to the swap. The file system chosen for the Linux is ext2, the one used for Windows 2000 is NTFS.

Back

APPENDIX C

During the stage we could use and find informations on several books:

- **"The c programming language" (2nd edition) by Kernighan and Ritchie;**
- **"Learning the VI editor" by Linda Lemb;**
- **"TCP/IP" by IBM's redbooks;**
- **"Programmazione dello Z80" by Rodnay Zacs;**

Back