

Computação com Software Livre

Eduardo Hiroshi Nakamura

12 de junho de 2023

Conteúdo

I	Comunicação de Dados	2
1	Introdução	3
2	Common Gateway Interface	6
2.1	Common Gateway Interface	6
II	Compiladores	15
3	Introdução	16
3.1	Processo de Compilação	16
4	Análise léxica	17
4.1	Análise Léxica	17
4.1.1	Automato Finito Determinístico	17
4.1.2	Expressão Regular	18
5	Análise sintática	19
5.1	Análise Sintática	19
5.1.1	Gramática	19
5.1.2	Ambiguidade	21
5.1.3	Recursão à esquerda	21
III	Segurança de Computadores	24
6	Criptografia	25
6.1	Introdução	25
6.2	Criptografia Simétrica	26
6.2.1	Cifra de César	26
6.3	Criptografia Assimétrica	26
6.3.1	RSA	26

Parte I

Comunicação de Dados

Capítulo 1

Introdução

Os dados podem ser enviados de maneira Paralela e Serial.

A comunicação de dados Paralela ocorre com todos os bits enviados pela via ao mesmo tempo. Se alguns dos bits atrasar ou adiantar a informação passada pelos dados é perdida. Esse problema de sincronização é um fator que pesa contra a vantagem da rapidez desse tipo de comunicação.

A comunicação de dados Serial os bits são enviados em sequência e a identificação do início e fim de cada dado é utilizado caracteres especiais.

A comunicação de dados tem por base usar a tabela ASCII para a transferência de dados. Então para contas numéricas é necessário converter o texto para valor numérico.

Listing 1.1: Table ASCII

```
#include <stdio.h>

int
main()
{
    char letra;
    int numero;

    printf("Entre com a numero\n");
    scanf("%c", &letra);
    numero = letra - 48;
    printf("A letra = %c\n", letra);
    printf("O numero convertido = %d\n", numero);
}
```

Para entender o programa a seguir é necessário sobre operações bit a bit e tabela ASCII.

Baud Rate ou Taxa de Transmissão e Porta(no código o /dev/pts/4) são parâmetros do ambiente da comunicação Serial.

Listing 1.2: Comunicação Serial

```

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <unistd.h>

int
main()
{
    int serial_port = open("/dev/pts/4", O_RDWR);
    struct termios tty;

    if(tcgetattr(serial_port, &tty) != 0)
    {
        printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
        return 1;
    }

    tty.c_cflag &= ~PARENB;
    tty.c_cflag &= ~CSTOPB;
    tty.c_cflag &= ~CSIZE;
    tty.c_cflag |= CS8;
    tty.c_cflag &= ~CRTSCTS;
    tty.c_cflag |= CREAD | CLOCAL;
    tty.c_lflag &= ~ICANON;
    tty.c_lflag &= ~ECHO;
    tty.c_lflag &= ~ECHOE;
    tty.c_lflag &= ~ECHONL;
    tty.c_lflag &= ~ISIG;
    tty.c_iflag &= ~(IXON | IXOFF | IXANY);
    tty.c_iflag &= ~(IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL);
    tty.c_oflag &= ~OPOST;
    tty.c_oflag &= ~ONLCR;
    tty.c_cc[VTIME] = 10;
    tty.c_cc[VMIN] = 0;
    cfsetispeed(&tty, B9600);
    cfsetspeed(&tty, B9600);

    if(tcsetattr(serial_port, TCSANOW, &tty) != 0)
    {
        printf("Error %i from tcsetattr: %s\n", errno, strerror(errno));
    }
}

```

```

return 1;
}

unsigned char msg[] = { 'H', 'e', 'l', 'l', 'o', '\r' };
write(serial_port, msg, sizeof(msg));

char read_buf [256];

memset(&read_buf, '\0', sizeof(read_buf));

int num_bytes = read(serial_port, &read_buf, sizeof(read_buf));

if(num_bytes < 0)
{
    printf("Error reading: %s", strerror(errno));
    return 1;
}

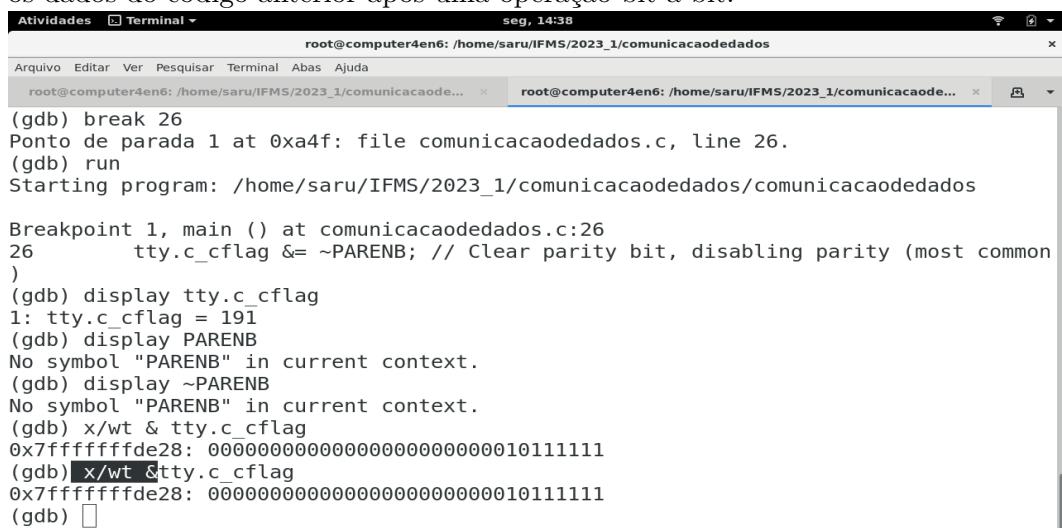
printf("Read %i bytes. Received message: %s", num_bytes, read_buf);

close(serial_port);
return 0;
}

```

A comunicação Serial é muito utilizada em configuração de Redes Industriais e Microncontroladores.

Na imagem a seguir um uso do GNU Debugger para visualizar em binário os dados do código anterior após uma operação bit a bit.



The screenshot shows a terminal window titled "Atividades Terminal" with the command "seg, 14:38". The window displays the following GDB session:

```

root@computer4en6: /home/saru/IFMS/2023_1/comunicacaodedados
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: /home/saru/IFMS/2023_1/comunicacaode... × root@computer4en6: /home/saru/IFMS/2023_1/comunicacaode... ×
(gdb) break 26
Ponto de parada 1 at 0xa4f: file comunicacaodedados.c, line 26.
(gdb) run
Starting program: /home/saru/IFMS/2023_1/comunicacaodedados/comunicacaodedados

Breakpoint 1, main () at comunicacaodedados.c:26
26          tty.c_cflag &= ~PARENB; // Clear parity bit, disabling parity (most common)
(gdb) display tty.c_cflag
1: tty.c_cflag = 191
(gdb) display PARENB
No symbol "PARENB" in current context.
(gdb) display ~PARENB
No symbol "PARENB" in current context.
(gdb) x/wt & tty.c_cflag
0x7fffffffde28: 0000000000000000000000000000000010111111
(gdb) x/wt & tty.c_cflag
0x7fffffffde28: 0000000000000000000000000000000010111111
(gdb) 

```

Capítulo 2

Common Gateway Interface

2.1 Common Gateway Interface

Common Gateway Interface ou CGI é a tecnologia atrás das páginas dinâmicas de internet. Para se ter um programa em CGI uma linguagem de programação precisa ter as seguintes opções: Ler variáveis de ambiente, Imprimir mensagens e Trabalhar com arquivos.

O CGI trabalha em cima de um servidor do protocolo HTTP(Hypertext Transfer Protocol) de páginas de internet, como o Apache.

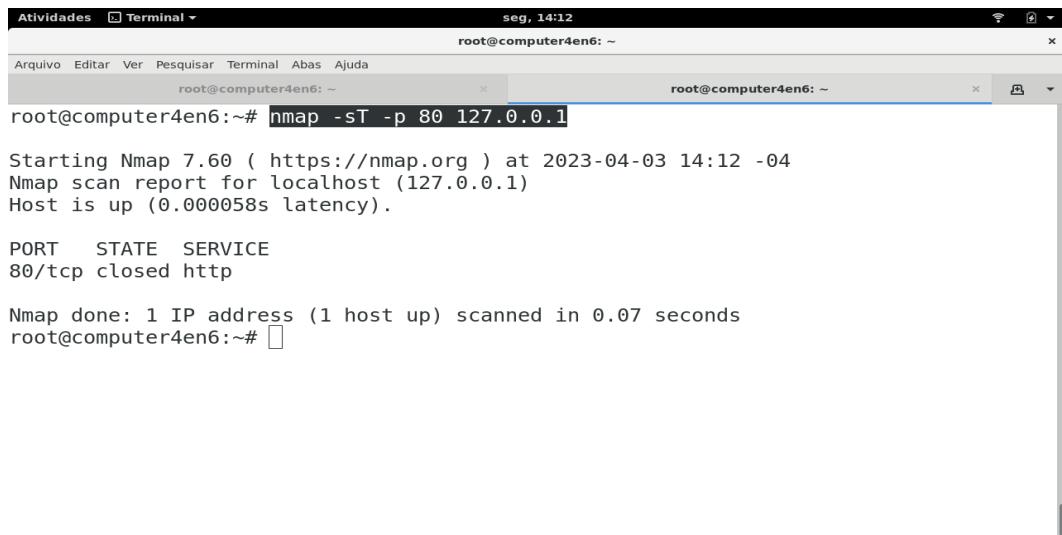
Na figura a seguir a instalação do Apache e do NMAP(Network Mapper).



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has two tabs, both titled 'root@computer4en6: ~'. The left tab shows the command 'root@computer4en6:~# su -' followed by the output of the 'apt-get install apache2 nmap' command. The output indicates that Nmap is already the newest version (7.60-1ubuntu5) and Apache2 is already the newest version (2.4.29-1ubuntu4.27). It also lists packages that were automatically installed and are no longer needed, such as linux-hwe-5.4-headers-5.4.0-131 through 139. The right tab of the terminal window is currently inactive.

```
Atividades Terminal seg, 14:11
root@computer4en6: ~
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: ~
root@computer4en6:~# su -
root@computer4en6:~# apt-get install apache2 nmap
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
nmap is already the newest version (7.60-1ubuntu5).
apache2 is already the newest version (2.4.29-1ubuntu4.27).
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  linux-hwe-5.4-headers-5.4.0-131 linux-hwe-5.4-headers-5.4.0-132
  linux-hwe-5.4-headers-5.4.0-135 linux-hwe-5.4-headers-5.4.0-136
  linux-hwe-5.4-headers-5.4.0-137 linux-hwe-5.4-headers-5.4.0-139
Utilize 'apt autoremove' para os remover.
```

Na figura a seguir NMAP varrendo a porta 80 do protocolo HTTP e verificando que o servidor Apache está executando com a porta aberta no Linux .

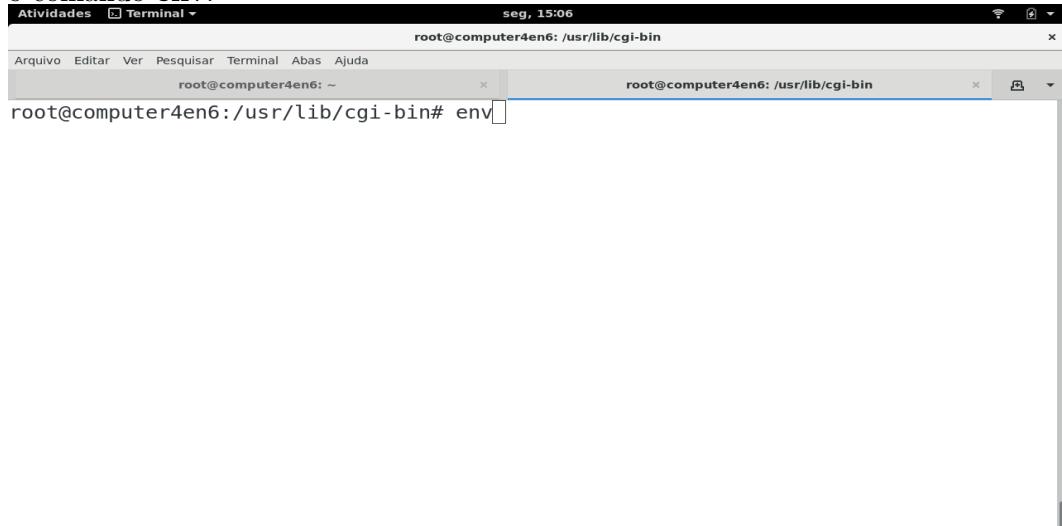


```
Atividades Terminal seg, 14:12
root@computer4en6: ~
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: ~
root@computer4en6:~# nmap -sT -p 80 127.0.0.1
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-03 14:12 -04
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000058s latency).

PORT      STATE SERVICE
80/tcp    closed http

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
root@computer4en6:~#
```

Na figura a seguir conseguimos ver as variáveis de ambiente no Linux com o comando **env**.



```
Atividades Terminal seg, 15:06
root@computer4en6: /usr/lib/cgi-bin
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: ~
root@computer4en6: /usr/lib/cgi-bin
root@computer4en6:/usr/lib/cgi-bin# env
```

No código a seguir um programa básico em CGI.

Listing 2.1: Primeiro programa CGI

```
#include <stdio.h> // printf()

int
main()
{
    printf("Content-type:text/html\n\n");
    printf("Programando por um mundo melhor\n");
```

```

return (0);
}

```

No código a seguir a impressão das variáveis de ambiente CGI.

Listing 2.2: Variáveis de ambiente CGI

```

#include <stdio.h>

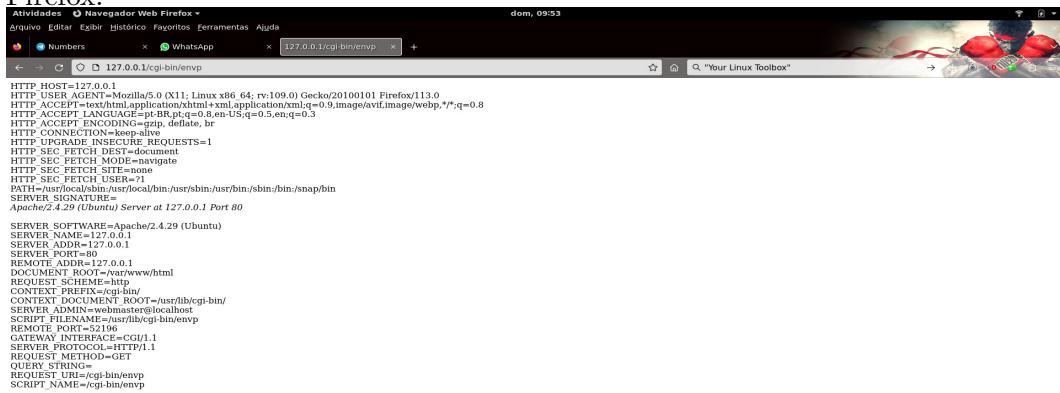
int
main(int argc, char **argv, char **envp)
{
    printf( "Content-type:text/html\n\n" );

    for (char **env = envp; *env != 0; env++)
    {
        char *variavelambiente = *env;

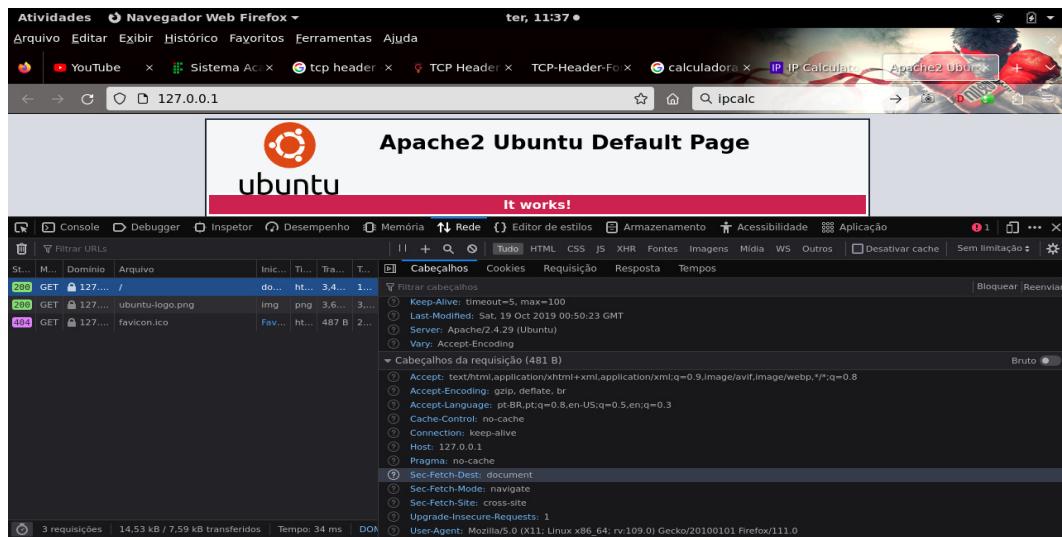
        printf( "%s<br>\n" , variavelambiente );
    }
    return (0);
}

```

Na figura a seguir conseguimos ver as variáveis de ambiente CGI no Mozilla Firefox.



Na figura a seguir conseguimos ver as variáveis de ambiente CGI através do Debugger do Mozilla Firefox.



Listing 2.3: Processamento da QUERYSTRING do CGI

```
#include <stdio.h> // printf()
#include <stdlib.h> // getenv(), EXIT_SUCCESS, atoi()
#include <string.h> // strtok_r()

int
main()
{
    char *str1;
    char *str2;
    char *token;
    char *subtoken;
    char *saveptr1;
    char *saveptr2;
    int i;
    int j;
    int valor;
    int total;

    printf("Content-type:text/html\n\n");

    total = 0;
    for(i = 1, str1 = getenv("QUERY_STRING"); ; i++, str1 = NULL)
    {
        token = strtok_r(str1, "&", &saveptr1);
        if(token == NULL)
        {
```

```

break;
}

j = 0;
for( str2 = token; ; str2 = NULL)
{
    subtoken = strtok_r(str2, "=" , &saveptr2);
    if(subtoken == NULL)
    {
        break;
    }
    if(j == 0)
    {
        printf( "%s\u00d7\n" , subtoken );
        j++;
    }
    else
    {
        valor = atoi(subtoken);
        printf( "%s<br>\n" , subtoken );
        total += valor;
        j = 0;
    }
}
printf( "<br>O\u00d7Total\u00d7da\u00d7soma\u00d7das\u00d7variaveis\u00d7%d\n" , total );

return(EXIT_SUCCESS);
}

```

Na figura a seguir resultado do processamento da variável de ambiente QUERYSTRING, conversão para inteiro e sua soma.



$v = 1$
 $w = 2$
 $x = 3$
 $y = 4$
 $z = 5$

O Total da soma das variaveis 15

No.	Time	Source	Destination	Protocol	Length
446	10.127.0.0.1		127.0.0.1	HTTP	551
448	10.127.0.0.1		127.0.0.1	HTTP	391

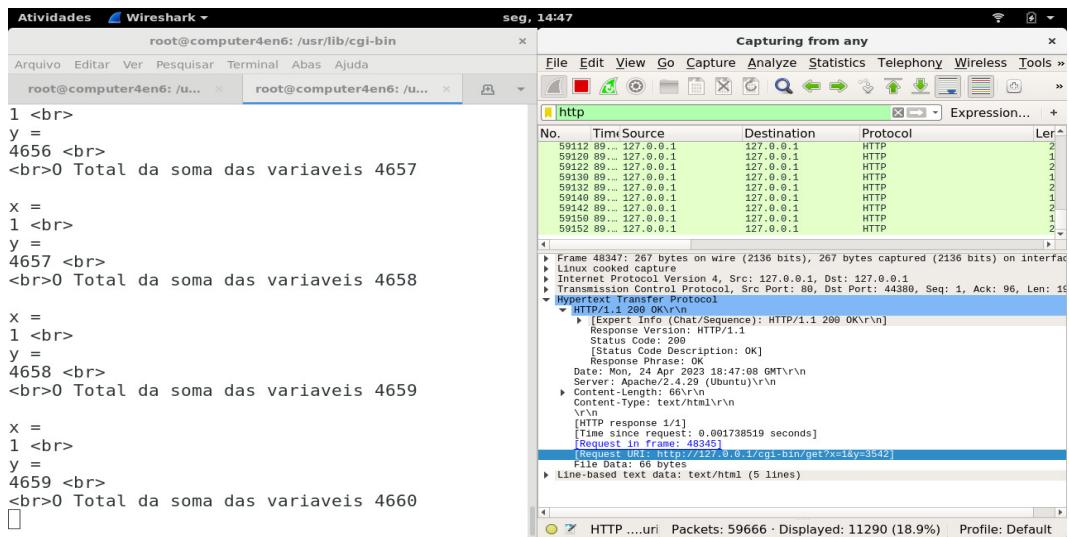
Na figura a seguir a utilização do programa CURL no acesso ao CGI via linha de comando.

```
Atividades Terminal seg, 14:38
root@computer4en6: /usr/lib/cgi-bin
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: /usr/lib/cgi-bin
root@computer4en6: /usr/lib/cgi-bin
root@computer4en6: /usr/lib/cgi-bin# curl "http://127.0.0.1/cgi-bin/get?x=1&y=10"
x =
1 <br>
y =
10 <br>
<br>0 Total da soma das variaveis 11
root@computer4en6: /usr/lib/cgi-bin#
```

Na figura a seguir a automação do acesso da CGI pelo CURL através do poder da linha de comando.

```
Atividades Terminal seg, 14:43
root@computer4en6: /usr/lib/cgi-bin
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: /usr/lib/cgi-bin
root@computer4en6: /usr/lib/cgi-bin
root@computer4en6: /usr/lib/cgi-bin# for y in {1..10}; do curl "http://127.0.0.1/cgi-bin/get?x=1&y=$y" && echo " " ; done
x =
1 <br>
y =
1 <br>
<br>0 Total da soma das variaveis 2
x =
1 <br>
y =
2 <br>
<br>0 Total da soma das variaveis 3
x =
1 <br>
y =
3 <br>
<br>0 Total da soma das variaveis 4
x =
1 <br>
y =
4 <br>
```

Na figura a seguir visualizando o processo de automação através do sniffer Wireshark.



Um endereço de página de internet ou URL(Uniform Resource Locator) possui o seguinte esquema:

`http://usuario:senha@dominio:porta/caminhodediretorios/arquivo?variavel1=valor1&variavel2=valor2&variavel3=valor3 ...`

- usuario - opcional
- senha - obrigatório se for inserido usuario
- dominio - obrigatório, pode se substituído pelo número do IP(Internet Protocol).
- porta - opcional
- caminhodediretorio - obrigatório se existir
- arquivo - obrigatório
- variaveis - obrigatório se existir
- variaveis - obrigatório se for inserido variáve;

Os caracteres @, :, ? e & são codificados quando enviados em CGI para não confusão no seu processamento.

Na figura a seguir os caracteres especiais do CGI convertidos para ASCII na representação hexadecimal.



Parte II

Compiladores

Capítulo 3

Introdução

3.1 Processo de Compilação

A função de um compilador é transformar um código fonte em um programa executável.

Código Fonte	↔	
↓		
Pré-processado	↔	arquivo.c
↓		↓
Assembly	↔	arquivo.i
↓		↓
Binário reallocável	↔	arquivo.s
↓		↓
Executável	↔	arquivo.o
		↓
		arquivo

Na Tabela 3.1 acima temos as etapas em que o compilador GCC transforma um código fonte em C em um executável do Linux.

Na primeira etapa o código fonte é pré-processado e os arquivos de cabeçalho são incluídos dentro do código fonte o qual fez sua inclusão e as macros desse arquivo fonte são expandidas.

Na segunda etapa o processo é dividido em 3 etapas sequenciais: Analisador Léxico, Analisador Sintático e Analisador Semântico.

O Analisador Léxico identifica e emite erros relacionados as palavras permitidas em uma linguagem de programação.

O Analisador Sintático verifica a correta sequência entre as palavras e emite erros relacionados as regras gramaticais de uma linguagem de programação.

O Analisador Semântico verifica o correto relacionamento entre as partes do código fonte. Por exemplo: verificação de tipos.

Capítulo 4

Análise léxica

4.1 Análise Léxica

Um Analisador Léxico identifica tokens em um código fonte e seus respectivos lexemas e os salva em uma tabela de símbolos.

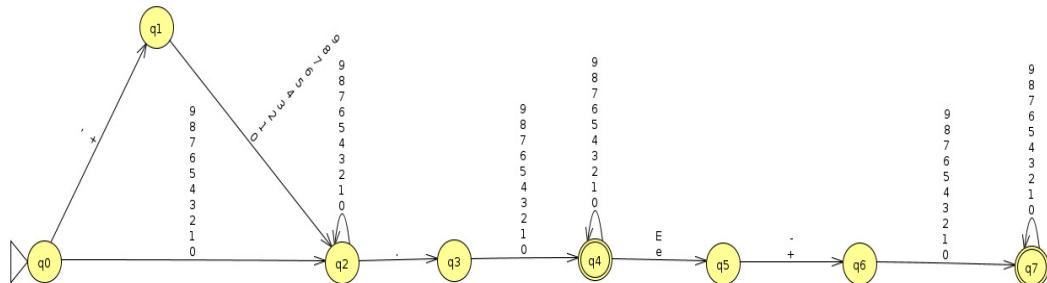
Token é a forma de identificar um palavra ou símbolo válido em uma linguagem de programação. O Lexema é a palavra ou símbolo indentificado pelo Token.

A tabela de símbolos é utilizada em todos os processos seguintes.

A implementação do reconhecimentos dos tokens é feita atravé das expressões regulares para identificar seus padrões. E essas definidas por Autômatos Finito Determinísticos.

4.1.1 Automato Finito Determinístico

Automato Finito Determinístico para validação do tipo float em C.



4.1.2 Expressão Regular

Expressão Regular correspondente ao Automato Finito Determinístico para o tipo float em C.

The screenshot shows the regex101.com interface. The URL in the address bar is `https://regex101.com`. The search term in the search bar is "expressão regular". The regular expression input field contains the pattern `^ / [\u00b1 | -] ? [0-9] * \. [0-9] + ([e | E] [\u00b1 | -] [0-9] +) ? $`. The results section shows 16 matches from a test string containing various floating-point numbers. The "EXPLANATION" section provides details about the regex components, such as the use of `\u00b1` and `e` for scientific notation. The "MATCH INFORMATION" section lists two matches: Match 1 at index 0-3 with value 1.0 and Match 2 at index 4-8 with value 3.14. The "QUICK REFERENCE" section lists common tokens and their meanings.

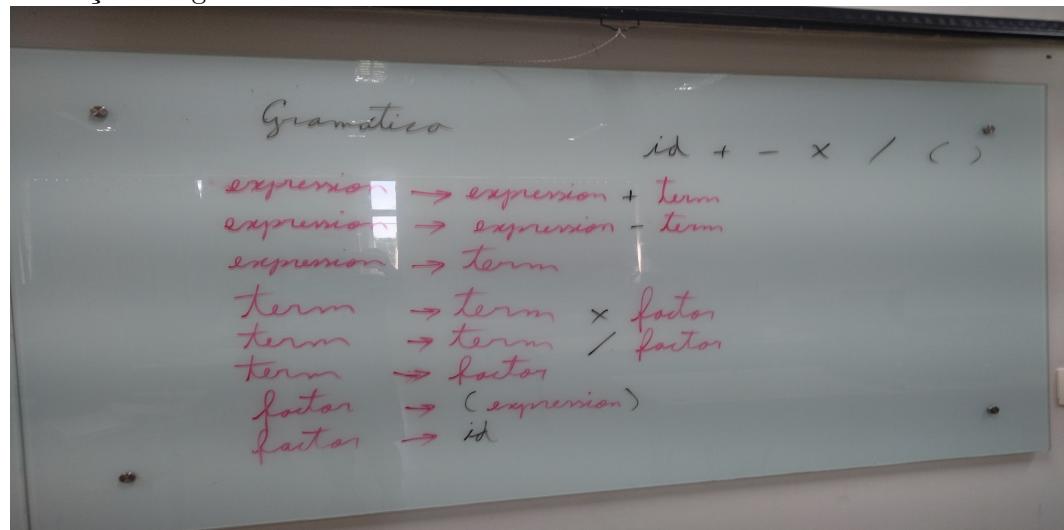
Capítulo 5

Análise sintática

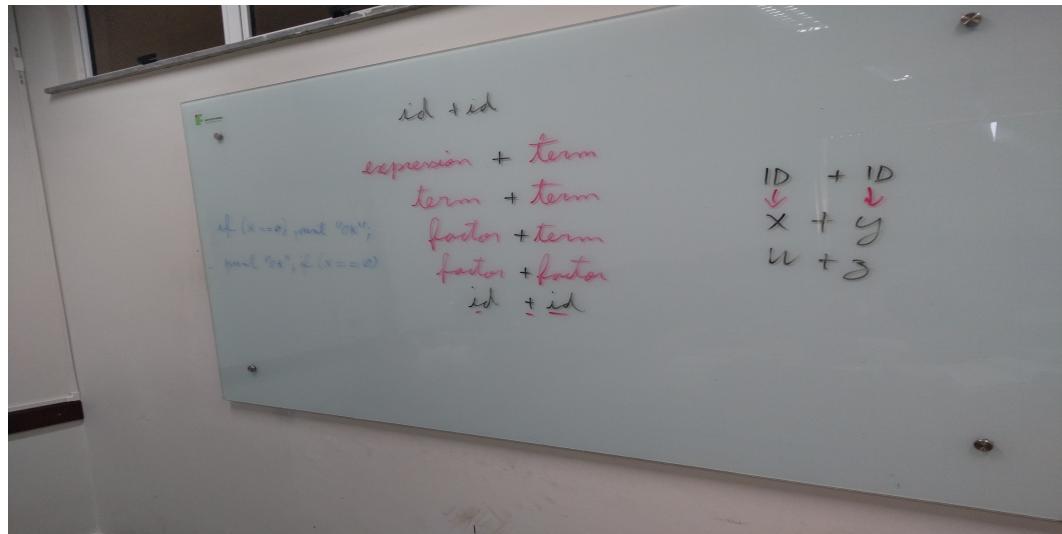
5.1 Análise Sintática

5.1.1 Gramática

Produções da gramática.

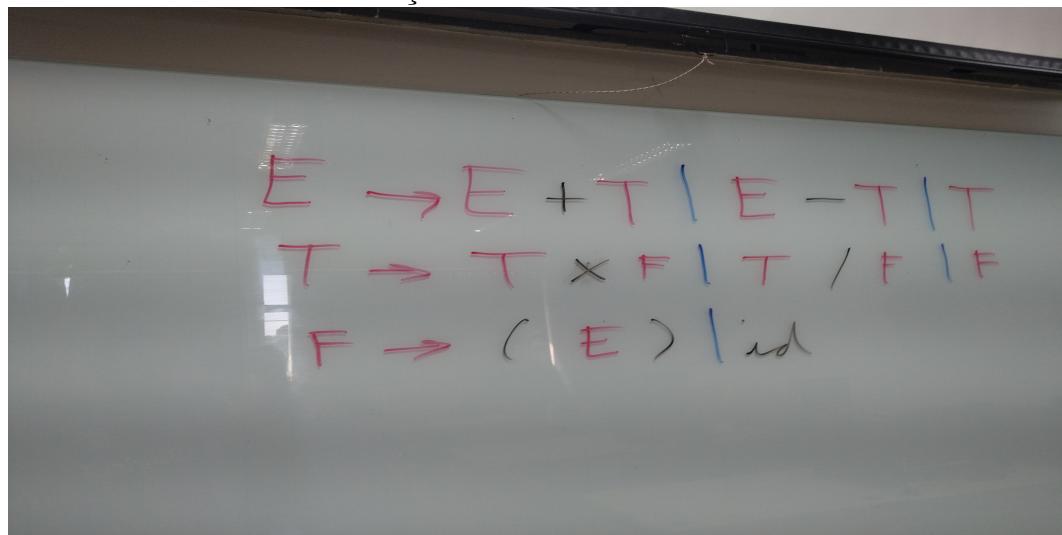


A esquerda da seta os **não terminais** e a direita sua produções.



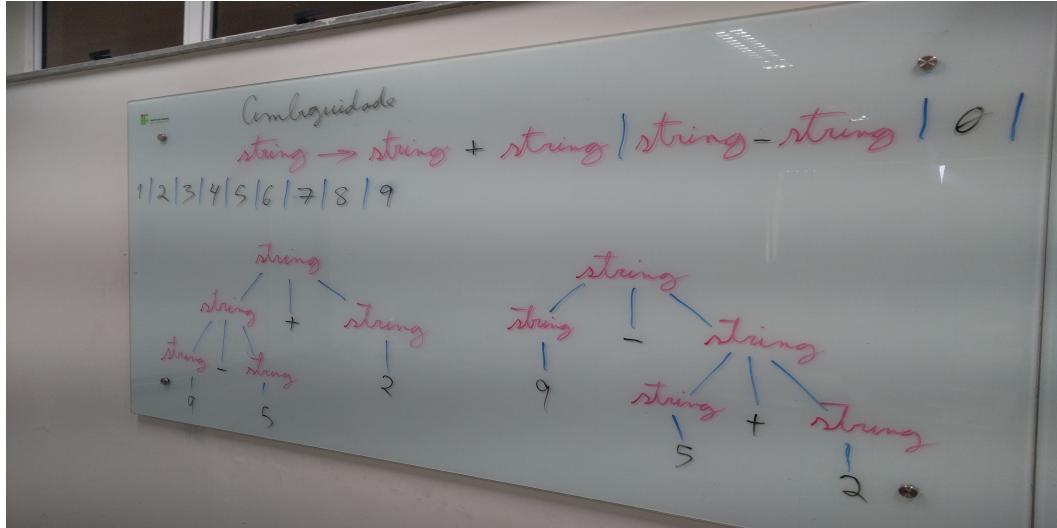
Comumente os **não terminais** são representados por letras maiúsculas e os **terminais** por minúsculas.

Os não terminais podem ser substituídos por eles mesmos e por terminais.
Esse último encerra as substituições.

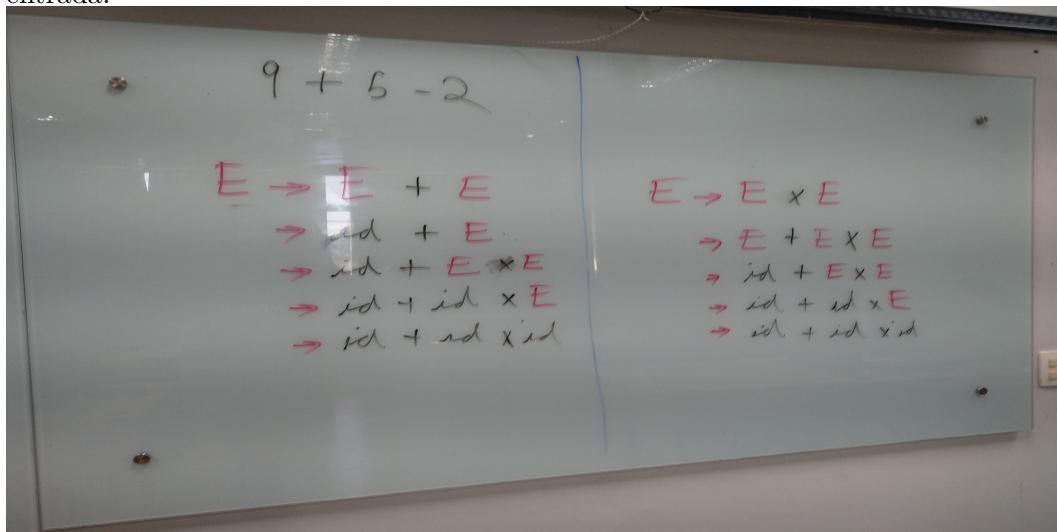


5.1.2 Ambiguidade

Gramática ambígua.

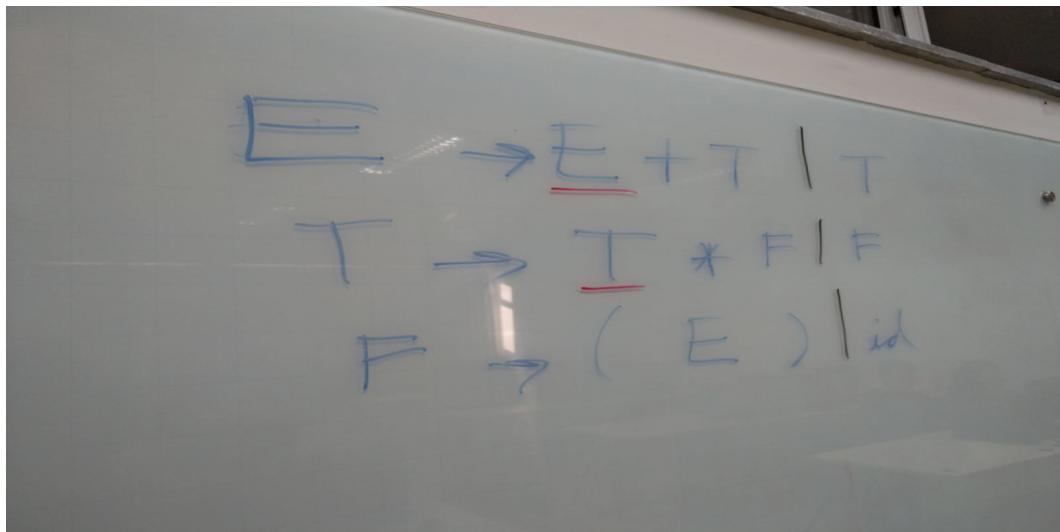


Uma gramática é ambígua quando gera 2 ou mais árvores para uma mesma entrada.

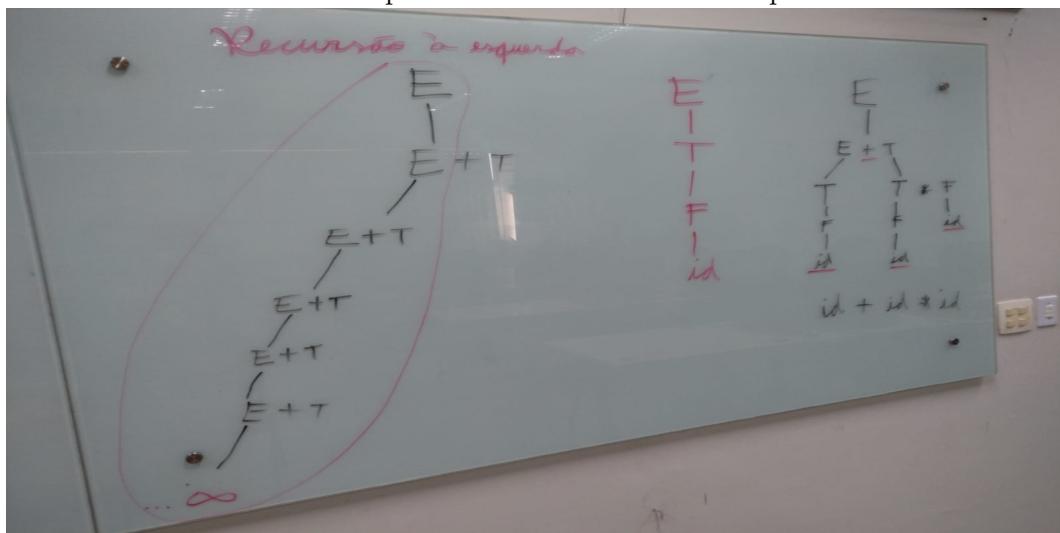


5.1.3 Recursão à esquerda

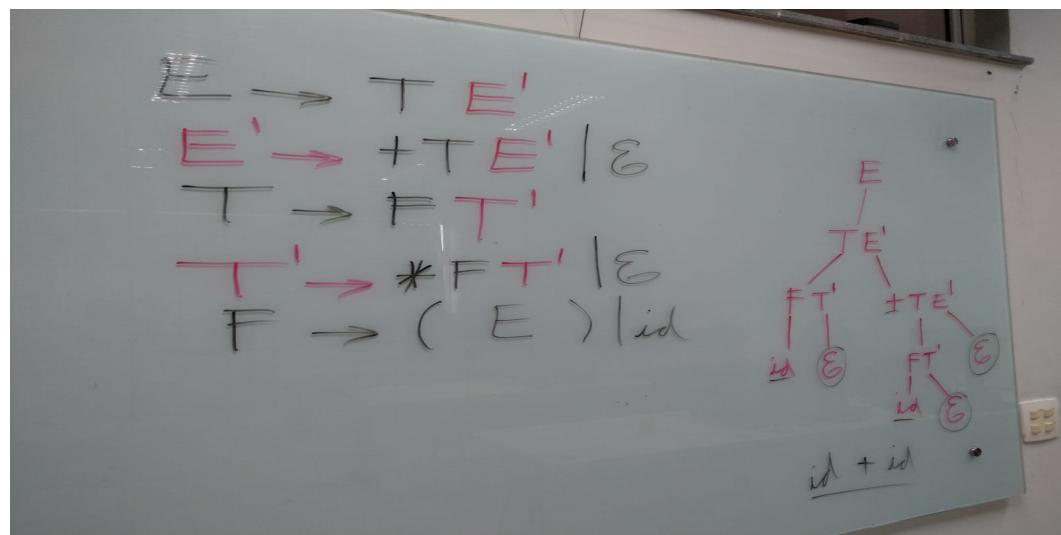
A gramática da figura a seguir possui uma recursão à esquerda, pois em sua produções um não terminal pode ser substituído por ele mesmo no primeiro termo da sua produção que é iniciada pela esquerda.



A recursão à esquerda pode gerar uma árvore com a primeira da esquerda da figura a seguir. A segunda e terceira árvores também podem ser geradas, mas esse indeterminismo não pode ser colocado em um compilador.



Para a remoção da recursão à esquerda é necessário criar novas produções derivadas da originais. Cada nova produção inicia-se com o terminal da produção original seguido do não terminal que estava à direita do original e o novo terminal. Ou a nova produção se é encerra em ε . A produção original também é modificada, ela inicia-se com o não terminal que estava a direita seguido do símbolo da nova produção. Na figura a seguir gramática com a remoção da recursão à esquerda e uma árvore com a mesma sentença que gera recursão à esquerda na gramática original.



Parte III

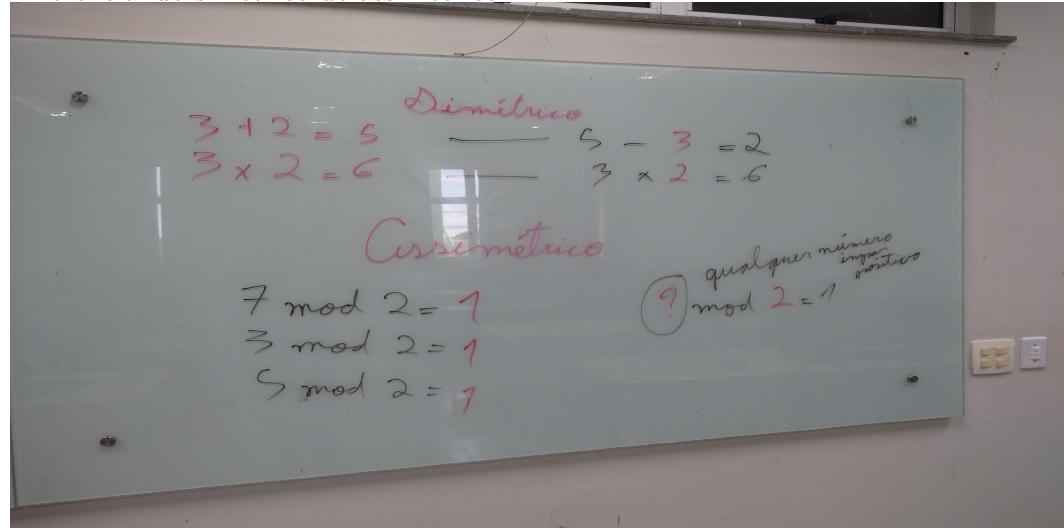
Segurança de Computadores

Capítulo 6

Criptografia

6.1 Introdução

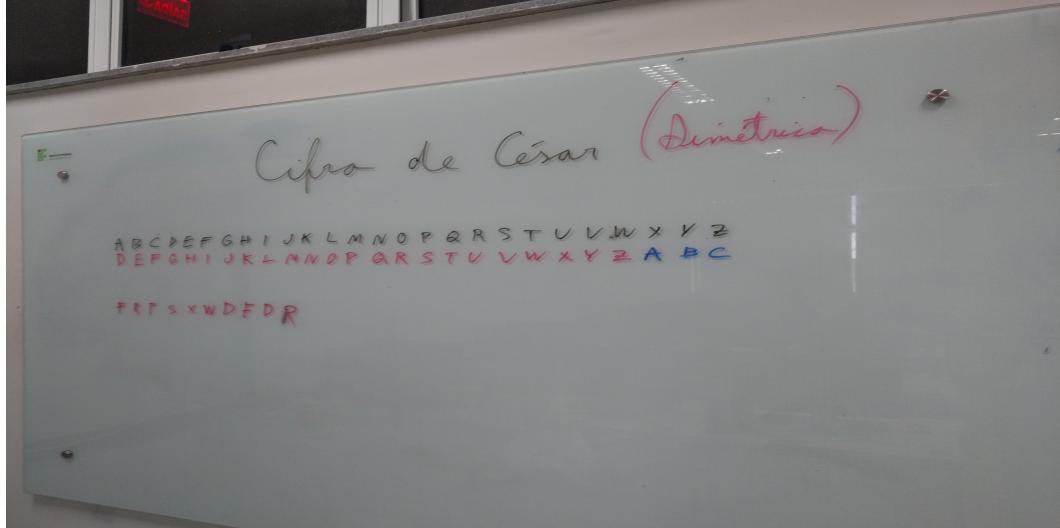
Diferenciando simétrico de assimétrico.



6.2 Criptografia Simétrica

6.2.1 Cifra de César

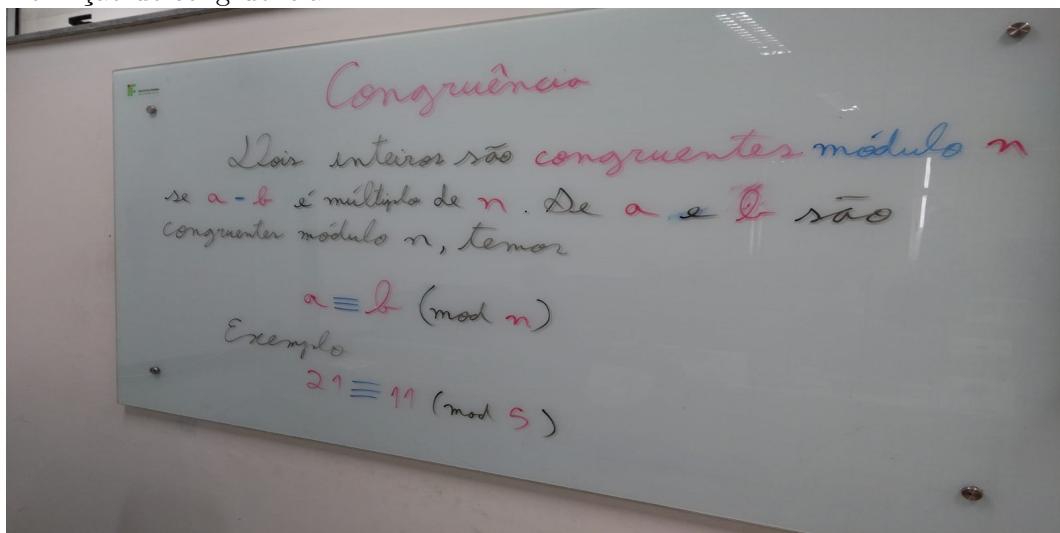
Cifra de César com deslocamento 3.



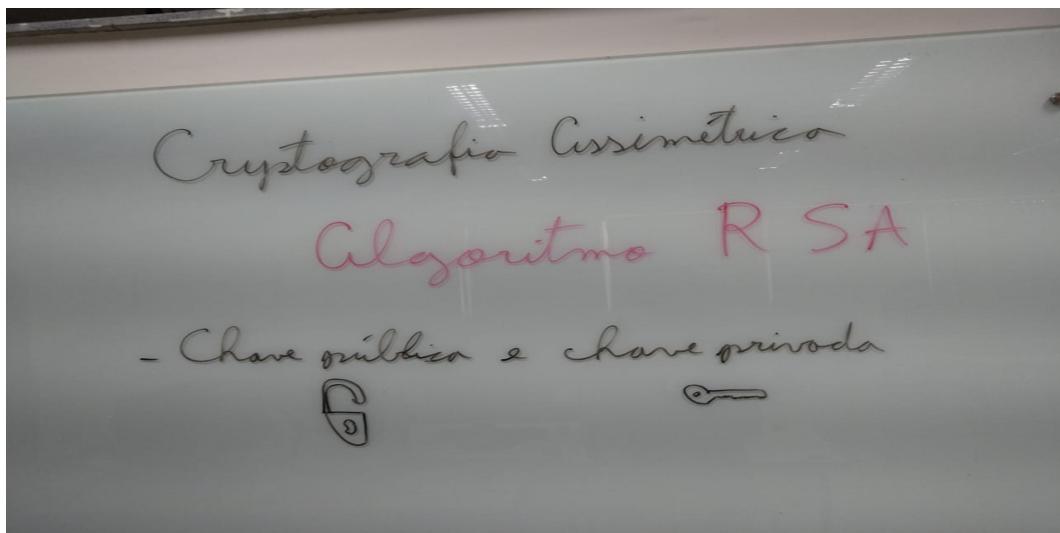
6.3 Criptografia Assimétrica

6.3.1 RSA

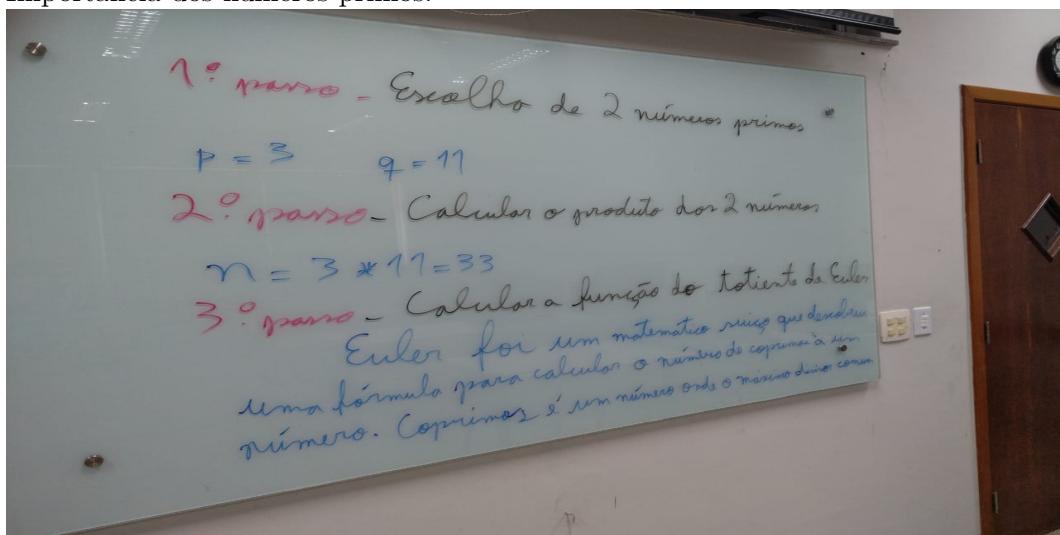
Definição de congruência.



Distinção entre chave pública e privada.



Importância dos números primos.



Cálculo do totiente.

entre ele e os outros números é 1.

$$\text{totiente}(n) = (p-1) * (q-1)$$

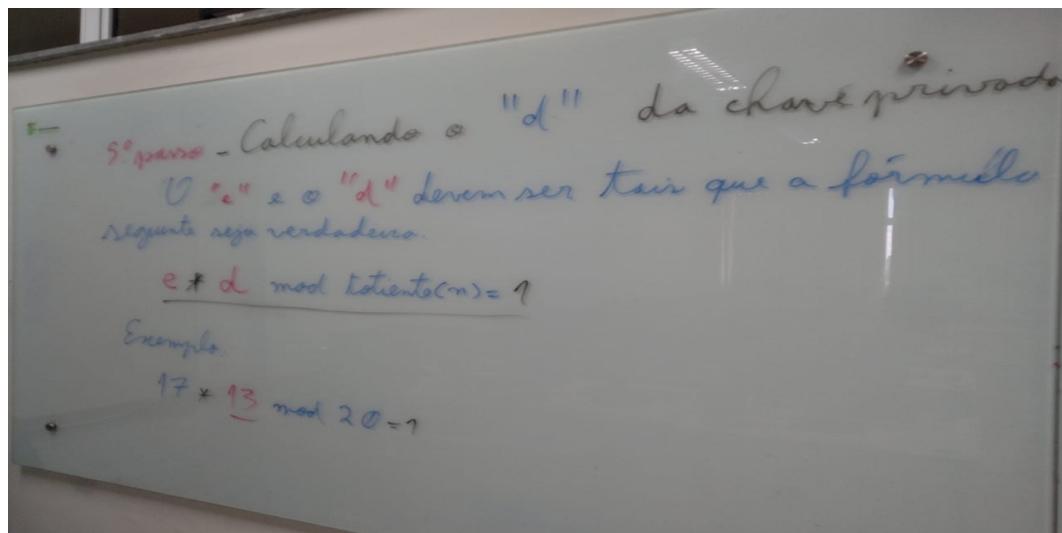
$$\text{totiente}(33) = (3-1) * (11-1) = 2 * 10 = 20$$

$MDC(33, 1) = 1$	$MDC(33, 11) = 11$	$MDC(33, 2) = 1$	$MDC(33, 22) = 11$
$MDC(33, 2) = 1$	$MDC(33, 12) = 3$	$MDC(33, 23) = 1$	$MDC(33, 32) = 1$
$MDC(33, 3) = 1$	$MDC(33, 13) = 1$	$MDC(33, 24) = 3$	$MDC(33, 37) = 1$
$MDC(33, 4) = 1$	$MDC(33, 14) = 1$	$MDC(33, 25) = 1$	$MDC(33, 38) = 1$
$MDC(33, 5) = 1$	$MDC(33, 15) = 3$	$MDC(33, 26) = 1$	$MDC(33, 39) = 1$
$MDC(33, 6) = 1$	$MDC(33, 16) = 1$	$MDC(33, 27) = 1$	$MDC(33, 40) = 1$
$MDC(33, 7) = 1$	$MDC(33, 17) = 1$	$MDC(33, 28) = 1$	
$MDC(33, 8) = 1$	$MDC(33, 18) = 3$	$MDC(33, 29) = 1$	
$MDC(33, 9) = 1$	$MDC(33, 19) = 1$	$MDC(33, 30) = 3$	
$MDC(33, 10) = 1$	$MDC(33, 20) = 1$		
$MDC(33, 11) = 1$			

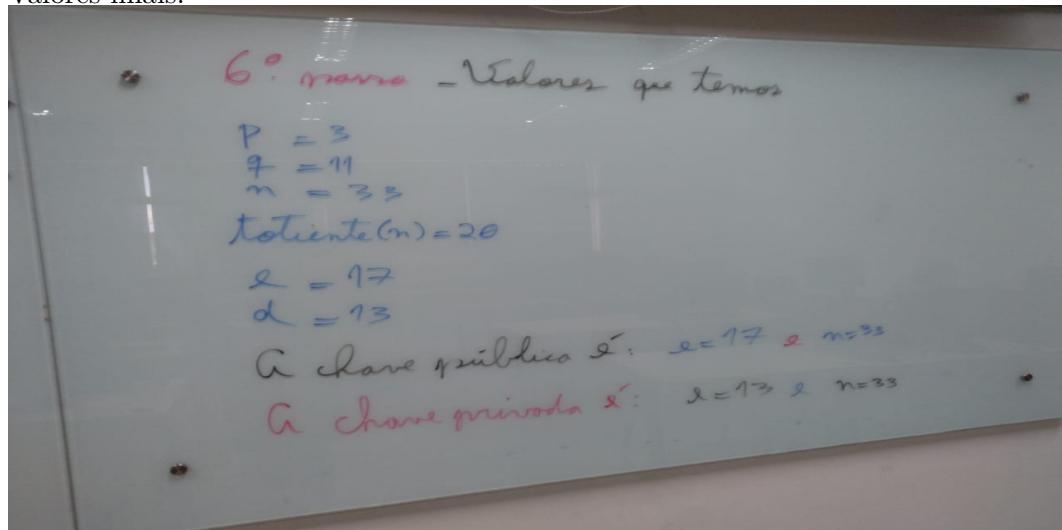
Números Coprimos e chave pública.

Os números Coprimos de 33 são:
1, 2, 4, 5, 7, 8, 10, 13, 14, 16, 17, 19, 20, 23, 25, 26, 28, 29, 31, 32
Totalizando 20.

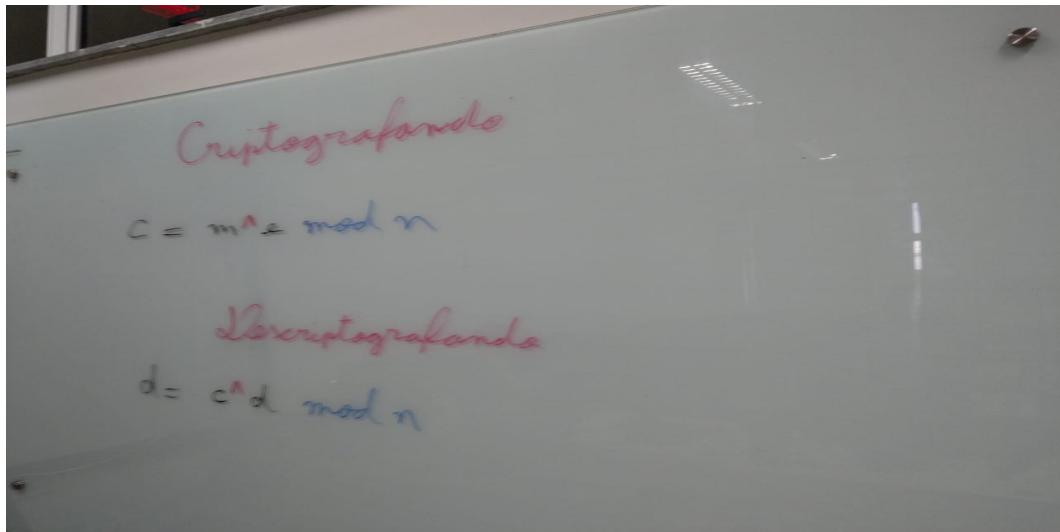
Chave privada.



Valores finais.



Fórmulas para criptografar e descriptografar.



Exemplo com uma mensagem.

```

Atividades Editor de texto v sex, 11:27
Abrir Salvar
cripto /home/saru/IFMS/2023_1/segurancadecomputadores
Criptografar
encrypt = m ^ e mod n
encrypt = 9 ^ 17 mod 33 = 16677181699666569 mod 33 = 15
Então nosso texto criptografado será 15
-----
Decriptografar
decrypt = c ^ d mod n
decrypt = 15 ^ 13 mod 33 = 1946195068359375 mod 33 = 9

```

Texto sem formatação Largura da tabulação: 8 Lin 8, Col 48 INS

Com o *scanner* NMAP é possível verificar o estado das portas de rede de um computador. Os estados das portas podem ser: aberto, fechado, filtrada ou bloqueada. Na figura a seguir o NMAP verificando o estado da porta 80 do protocolo TCP.

```

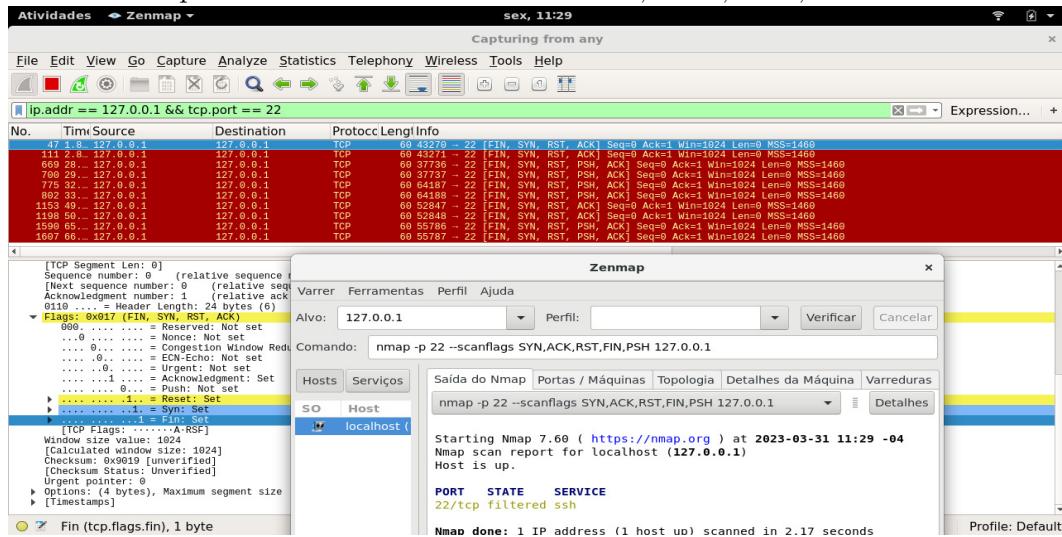
Atividades Terminal seg, 14:12
root@computer4en6: ~
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: ~
root@computer4en6:~# nmap -ST -p 80 127.0.0.1
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-03 14:12 -04
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000058s latency).

PORT      STATE SERVICE
80/tcp    closed http

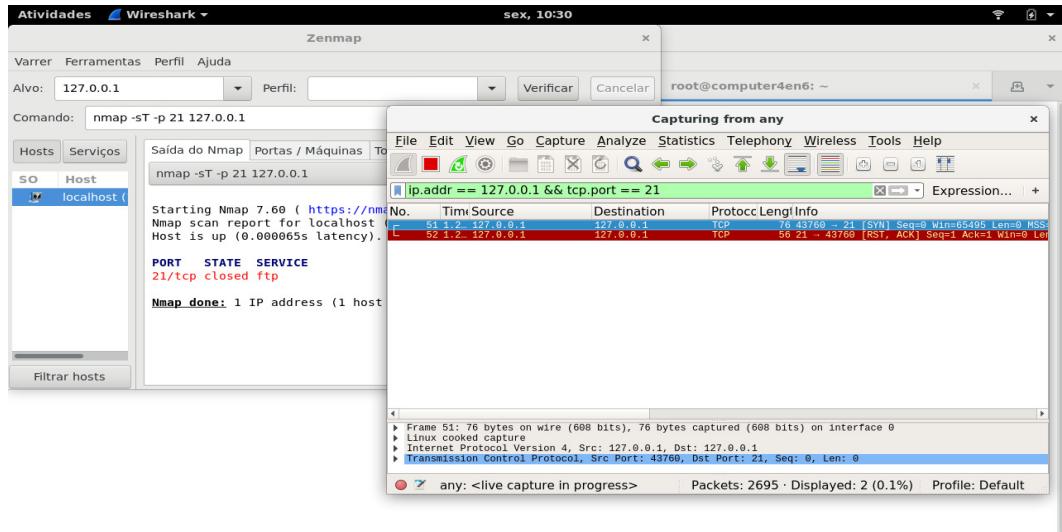
Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
root@computer4en6:~# 

```

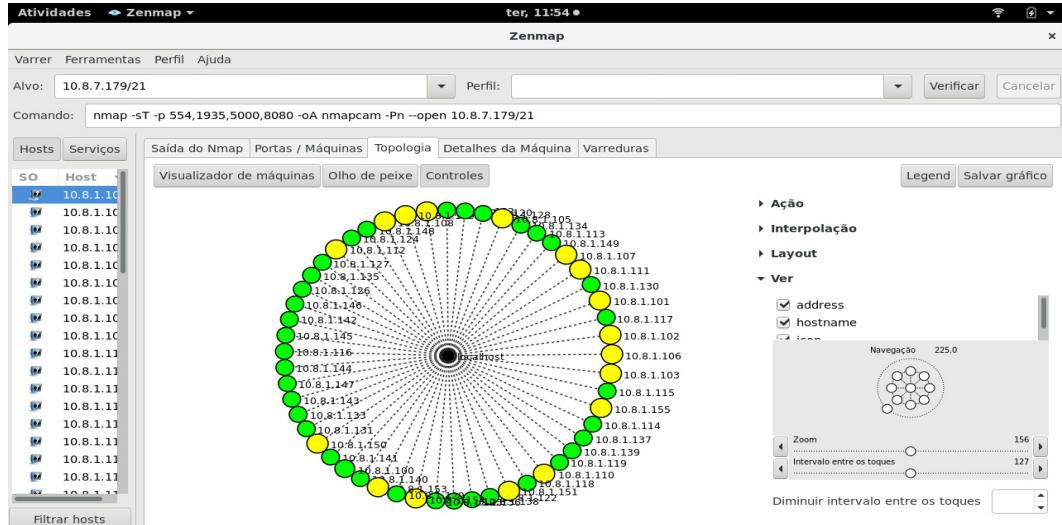
O *sniffer* Wireshark permite fazer o *debug* ou depuração dos pacotes de rede visualizando-os um a um, inclusive permitindo a filtragem deles. Na figura a seguir o Wireshark filtrando apenas pacotes vindos do IP 127.0.0.1 e da porta 22 TCP. Nessa imagem há o uso em conjunto com o NMAP que está verificando a porta 22 com as TCP FLAGS: SYN, ACK, RST, FIN e PSH.



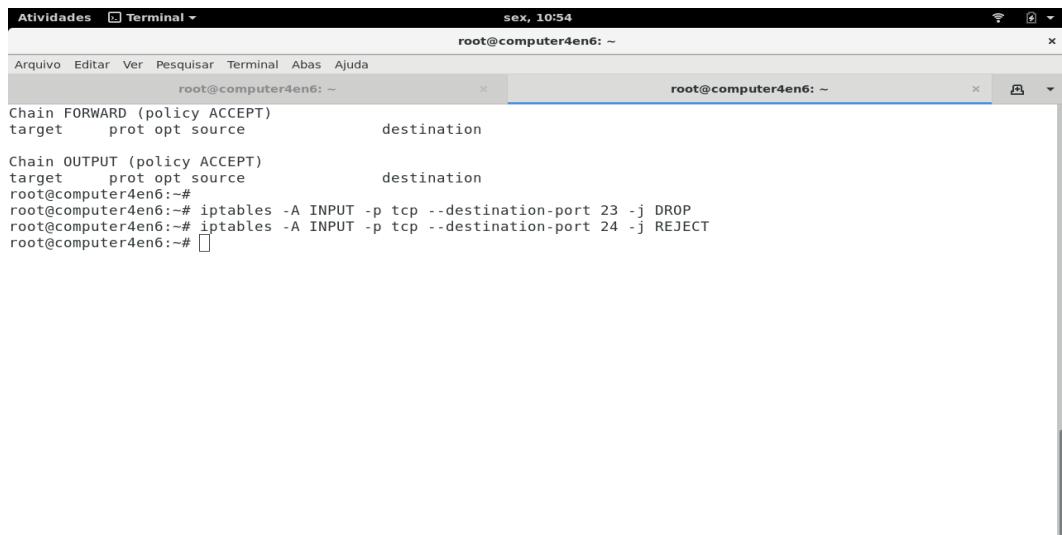
Na figura a seguir novamente o uso em conjunto do Scanner e do Sniffer, nela é detectado uma porta fechada e as respectivas TCP FLAGS de quando uma porta está fechada.



O Zenmap é uma interface gráfica para o NMAP. Como podemos ver na figura a seguir ele permite nos gerar um gráfico da segurança dos computadores por ele verificado.



O IPTABLES é o software padrão do Linux para filtragem de pacotes em um sistema de firewall. Na figura a seguir ele rejeitando conexões na porta 24 TCP e bloqueando na porta 23 TCP. No bloqueio não há nenhuma resposta, na rejeição há uma resposta de retorno.



```

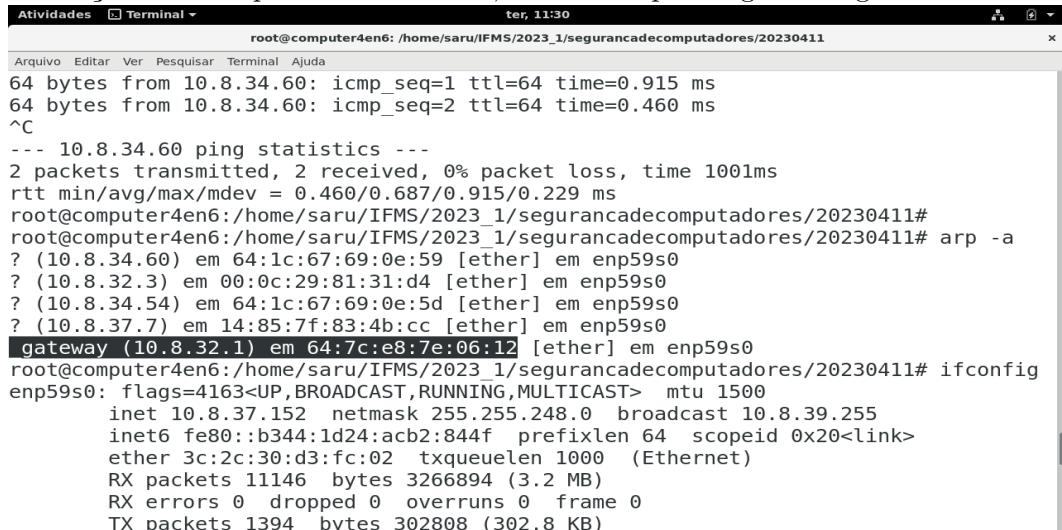
Atividades Terminal sexta, 10:54
root@computer4en6: ~

Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: ~ x
root@computer4en6: ~ x
root@computer4en6: ~ x

Chain FORWARD (policy ACCEPT)
target     prot opt source          destination
target     prot opt source          destination
root@computer4en6:~# iptables -A INPUT -p tcp --destination-port 23 -j DROP
root@computer4en6:~# iptables -A INPUT -p tcp --destination-port 24 -j REJECT
root@computer4en6:~# 

```

O comando ARP permite visualizar os últimos dispositivos que se comunicaram via rede ponto a ponto. Nessa visualização estão disponíveis o IP e o endereço MAC de quem se comunicou, como exemplo a figura a seguir.



```

Atividades Terminal ter, 11:30
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230411

Arquivo Editar Ver Pesquisar Terminal Ajuda
64 bytes from 10.8.34.60: icmp_seq=1 ttl=64 time=0.915 ms
64 bytes from 10.8.34.60: icmp_seq=2 ttl=64 time=0.460 ms
^C
--- 10.8.34.60 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.460/0.687/0.915/0.229 ms
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411#
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# arp -a
? (10.8.34.60) em 64:1c:67:69:0e:59 [ether] em enp59s0
? (10.8.32.3) em 00:0c:29:81:31:d4 [ether] em enp59s0
? (10.8.34.54) em 64:1c:67:69:0e:5d [ether] em enp59s0
? (10.8.37.7) em 14:85:7f:83:4b:cc [ether] em enp59s0
gateway (10.8.32.1) em 64:7c:e8:7e:06:12 [ether] em enp59s0
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# ifconfig
enp59s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.8.37.152 netmask 255.255.248.0 broadcast 10.8.39.255
          inet6 fe80::b344:1d24:acb2:844f prefixlen 64 scopeid 0x20<link>
            ether 3c:2c:30:d3:fc:02 txqueuelen 1000 (Ethernet)
              RX packets 11146 bytes 3266894 (3.2 MB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 1394 bytes 302808 (302.8 KB)

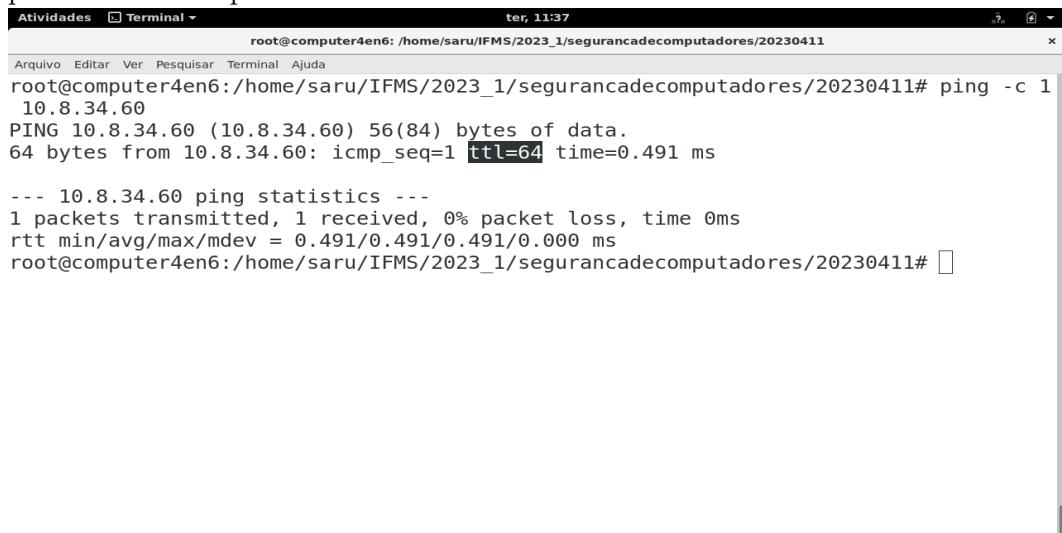
```

Na figura a seguir o comando *ifconfig* permite mudar o endereço MAC de um dispositivo de rede.



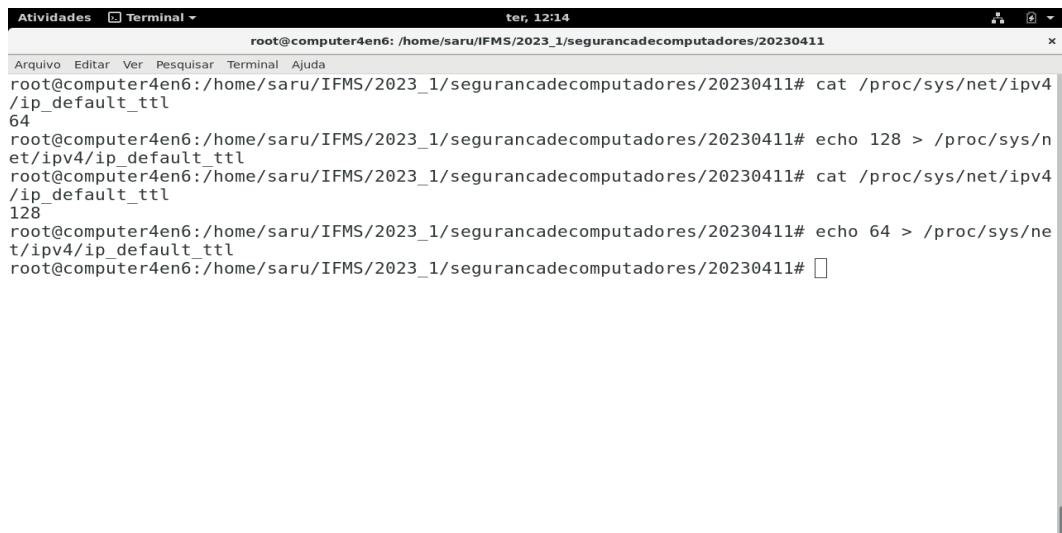
```
Atividades Terminal ter, 11:34
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230411
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# ifconfig
enp59s0  hw ether 64:1c:67:69:0e:5d
```

O TTL ou Time to Live é a quantidade de roteadores que um pacote admite passar antes que seja considerado esgotado. Na figura a seguir o TTL padrão de um dispositivo com Linux.



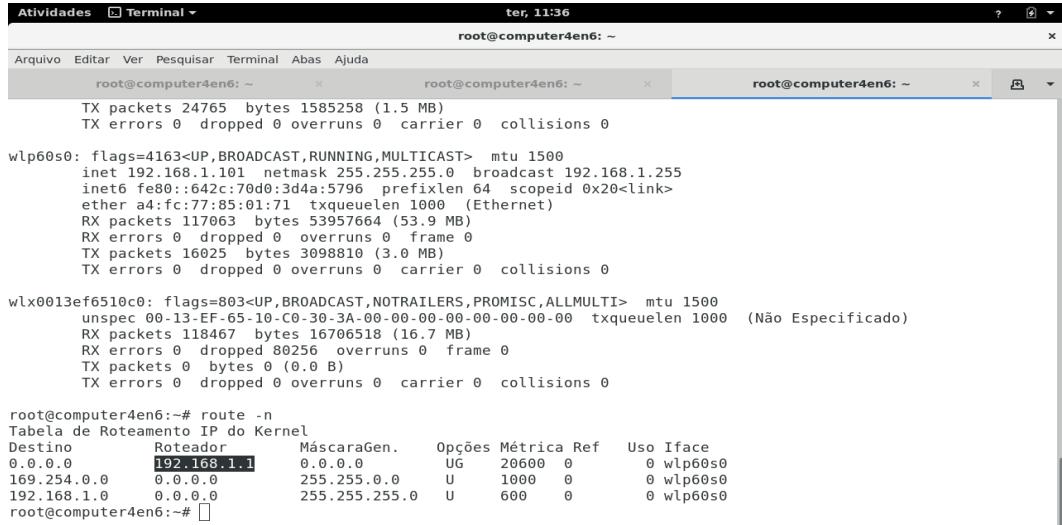
```
Atividades Terminal ter, 11:37
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230411
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# ping -c 1
10.8.34.60
PING 10.8.34.60 (10.8.34.60) 56(84) bytes of data.
64 bytes from 10.8.34.60: icmp_seq=1 ttl=64 time=0.491 ms
--- 10.8.34.60 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.491/0.491/0.491/0.000 ms
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411#
```

Na figura a seguir a mudança do TTL do Linux em tempo de execução.



```
Atividades Terminal ter, 12:14
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230411
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# cat /proc/sys/net/ipv4/ip_default_ttl
64
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# echo 128 > /proc/sys/net/ipv4/ip_default_ttl
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# cat /proc/sys/net/ipv4/ip_default_ttl
128
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411# echo 64 > /proc/sys/net/ipv4/ip_default_ttl
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230411#
```

O comando *route* permite verificar a rota que os pacotes terão conforme as faxas de IP. Na figura em destaque o GATEWAY para a rota que está sendo mostrada naquela linha.



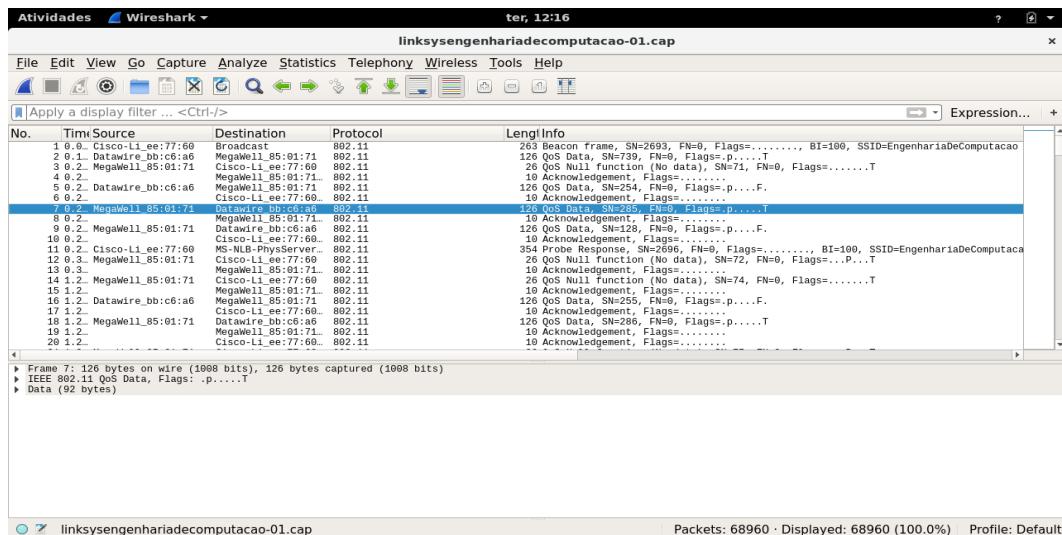
```
Atividades Terminal ter, 11:36
root@computer4en6: ~
Arquivo Editar Ver Pesquisar Terminal Abas Ajuda
root@computer4en6: ~          root@computer4en6: ~          root@computer4en6: ~
TX packets 24765 bytes 1585258 (1.5 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp60s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.101 netmask 255.255.255.0 broadcast 192.168.1.255
                ether a4:fc:77:85:01:71 txqueuelen 1000 (Ethernet)
                RX packets 117063 bytes 53957664 (53.9 MB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 16025 bytes 3098810 (3.0 MB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlx0013ef6510c0: flags=803<UP,BROADCAST,NOTRAILERS,PROMISC,ALLMULTI> mtu 1500
        unspec 00-13-EF-65-10-C0-30-3A-00-00-00-00-00-00 txqueuelen 1000 (Não Especificado)
                RX packets 118467 bytes 16706518 (16.7 MB)
                RX errors 0 dropped 80256 overruns 0 frame 0
                TX packets 0 bytes 0 (0.0 B)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@computer4en6:~# route -n
Tabela de Roteamento IP do Kernel
Destino      Roteador    MáscaraGen.  Opções Métrica Ref  Uso Iface
0.0.0.0      192.168.1.1  0.0.0.0      UG      20600  0      0 wlp60s0
169.254.0.0   0.0.0.0    255.255.0.0   U       1000   0      0 wlp60s0
192.168.1.0   0.0.0.0    255.255.255.0  U       600    0      0 wlp60s0
root@computer4en6:~#
```

O Wireshark consegue visualizar os dados de uma rede WIFI criptografada como vemos na figura a seguir.



Um HASH criptográfico é um cálculo feito em cima de um fluxo de bits. Um único bit diferente causa um HASH diferente. Na figura a seguir a demonstração com maiúsculas e minúsculas.

```
Atividades Terminal - root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428# echo "oi"
oi
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428# echo "oi" | md5sum
b9155515728fa0f69d9770f7877cb50a -
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428# echo "Oi" | md5sum
a9f73e97dc37ea6d242fc6e1aca0f15d -
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428# echo "oi" | md5sum
b9155515728fa0f69d9770f7877cb50a -
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428#
```

Na figura a seguir a demonstração que em um arquivo o HASH é feito pelo conteúdo do arquivo, não é considerado o nome dele. Arquivos copiados são identicos mesmo tendo nomes diferentes.

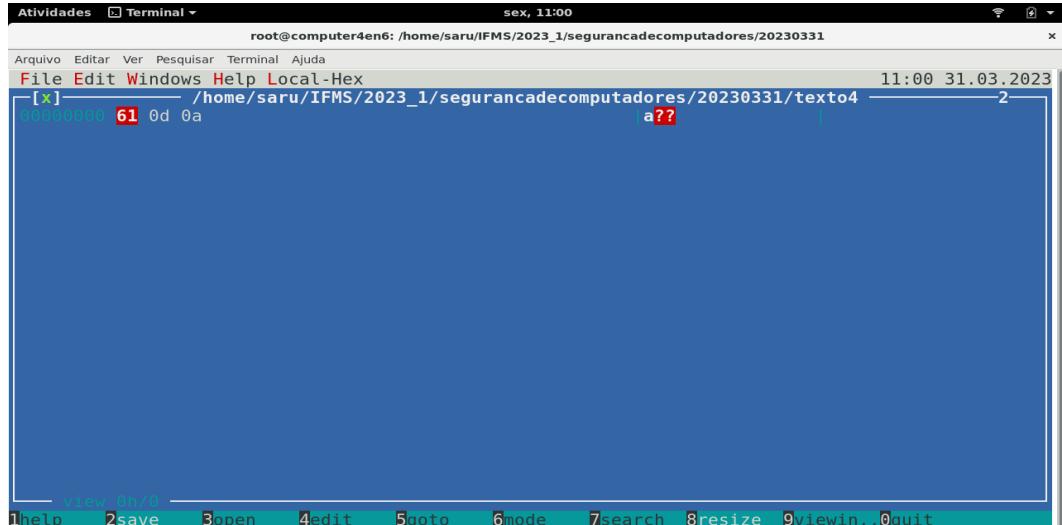


```

Atividades Terminal sex, 11:04
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428#
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# vim texto0
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# vim texto1
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# cp texto0 texto1
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# md5sum texto0 texto1
c8a8a19969fcf59d1351de2b5a3cd7c6  texto0
c8a8a19969fcf59d1351de2b5a3cd7c6  texto1
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# cp texto0 texto2
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# md5sum texto0 texto1 texto2
c8a8a19969fcf59d1351de2b5a3cd7c6  texto0
c8a8a19969fcf59d1351de2b5a3cd7c6  texto1
c8a8a19969fcf59d1351de2b5a3cd7c6  texto2
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# 

```

Em um editor de binários é possível observar que caracteres não imprimíveis fazem diferença, pois bits a mais alteram a conta e o resultado final do HASH criptográfico. Na figura a seguir um arquivo e seu final de linha.



```

Atividades Terminal sex, 11:00
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230331#
Arquivo Editar Ver Pesquisar Terminal Ajuda
File Edit Windows Help Local-Hex 11:00 31.03.2023
[x]-- /home/saru/IFMS/2023_1/segurancadecomputadores/20230331/texto4 --2
00000000 61 0d 0a |a??
1help 2save 3open 4edit 5goto 6mode 7search 8resize 9viewin.. 0quit

```

A figura a seguir apresenta dois arquivos com a mesma letra a em seu conteúdo, mas devido ao final de linha diferente o HASH criptográfico é diferente.



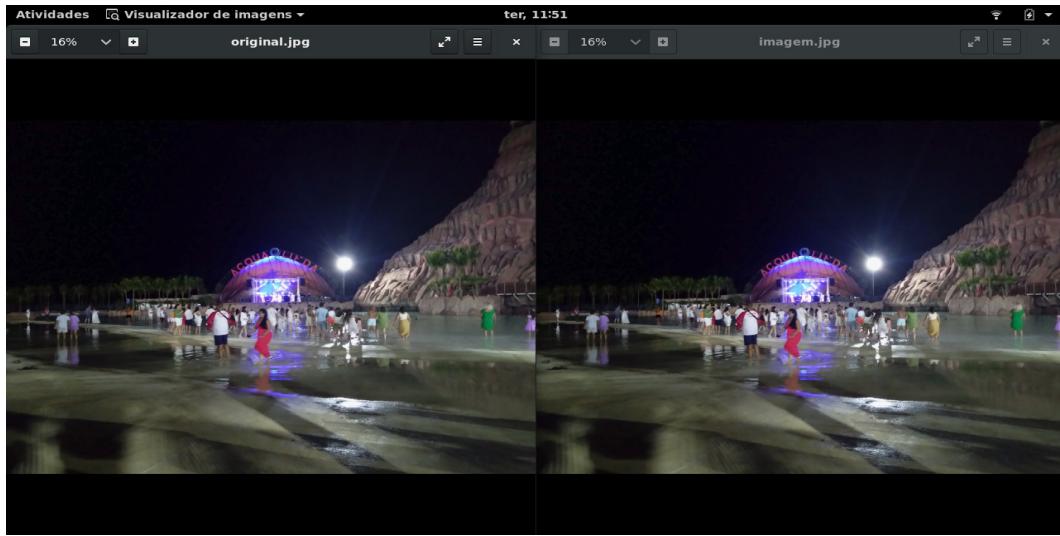
```
Atividades Terminal sexta, 11:08
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230428
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428# md5sum texto0 texto1 texto3
c8a8a19969fcbb59d1351de2b5a3cd7c6  texto0
01fdc44ef819db6273bc30965a23814  texto3
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230428#
```

Na figura a seguir o HASH criptográfico também aplicado a imagens com diferença em pixels. Um pixel não deixa de ser um dado salvo em bits, então muda o HASH.



```
Atividades Terminal ter, 11:43
root@computer4en6: /home/saru/IFMS/2023_1/segurancadecomputadores/20230502
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230502# md5sum *.png
b862de1e29808c0e1cbd639a0fecf18  3x3_H.png
c133d3c8d8126037df7fea6bd379ee0  3x3_modificado.png
25e3a9f5624282e27e7d308df9e83d20  3x3.png
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230502#
```

Nas duas figuras lado a lado a seguir mostram duas imagens visualmente muito semelhantes. Uma delas sofreu o processo de esteganografia, onde alguns pixels tiveram suas cores suavemente modificadas, para que nessa modificação sejam armazenadas informações. Essa modificação não é perceptível facilmente.



Na figura a seguir o HASH das duas imagens mostrando que elas são computacionalmente diferentes.

```

Atividades Terminal ter, 11:50
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputado... 0230502
Arquivo Editar Ver Pesquisar Terminal Ajuda
sagem.txt
Enter passphrase:
Re-Enter passphrase:
embedding "mensagem.txt" in "imagem.jpg"... done
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230502# md5sum *.jpg
bb759a8c191ce7a703300893ef39f67f imagem.jpg
cf92bdc478ee51851e58ab18aafee4ef original.jpg
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230502# 

```

The terminal window shows the command used to embed a message into an image and then verify the integrity using MD5 sums. The output indicates the message was successfully embedded and the files are different.

File browser view:

- original.jpg
- 3x3_H.c
- 3x3_H.png
- 3x3_H.xcf
- 3x3_modificado.c

Terminal window below:

```

root@computer4en6: ~
Pesquisar Terminal Ajuda
/opt/firefox/firefox
rror setting metrics feature config: JsonError("EOF while p
e: 1, column: 0)
ecoderStateMachine #1] WARNING: 7f9f9b0fedc0 OpenCubeb() fai
ile /builds/worker/checkouts/gecko/dom/media/AudioStream.cpp

```

Na figura a seguir o software de esteganografia extraiendo a mensagem da imagem.

```

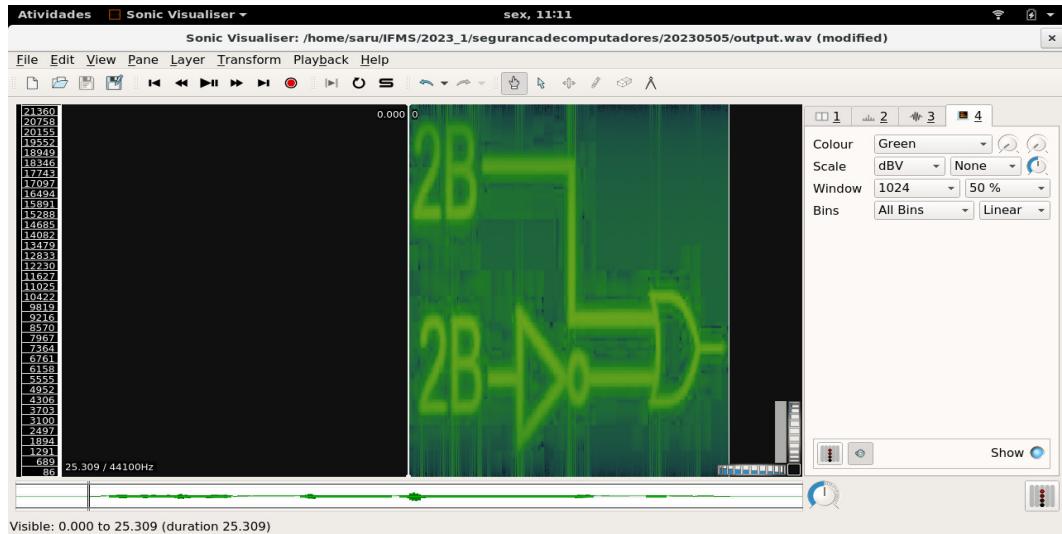
Atividades Terminal
mensagem.txt Salvar
Arquivo Editar Ver Pesquisar Terminal Ajuda
steghide: could not extract any data with that passphrase!
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230502# steghide extract -sf imagem.jpg
Enter passphrase:
wrote extracted data to "mensagem.txt".
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230502# 
```

Atráés de um software como o enscribe é possível transformar uma imagem em um áudio, esse processo é a criação de um marca d'água digital. Na figura esse processo.

```

Atividades Terminal
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230505
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230505# enscribe -ts=0 ifms-tl-marca-2015_90.png
Enscribe 0.1.0 (c) 2004-2008 Jason Downer
Homepage: http://jbd.zayda.net/enscribe/
This is ALPHA software.
Enscribe is free software, distributed under the General Public License.
There is NO warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.
Warning: No output filename, using output.wav instead
Encoding image ifms-tl-marca-2015_90.png into Microsoft WAV audio file output.wav
amplerate: 44100 Hz channels: 2
Scaling Image
Opening output file
Building output audio file
Done.
root@computer4en6:/home/saru/IFMS/2023_1/segurancadecomputadores/20230505# 
```

Na figura a seguir o software Sonic Visualizar que através de um espectrograma consegue visualizar a marca d'água digital.



O comando echo pode ser usado para gerar o HASH de um texto em linha de comando. Mas como é mostrado na figura a seguir é necessário passar o parâmetro -n senão ele insere um caractér de final de linha.

```

Atividades Terminal 6 de jun 11:19
oosaru@oosaru-G7-7588:/home/saru/IFMS/2023_1/segurancadecomputadores/SegundoTrabalho$ ./hashcriptograf
ico 1
INPUT: 1
SHA256: 4dff4ea340f0a823f15d3f4f01ab62eae0e5da579ccb851f8db9dfe84c58b2b37b89903a740e1ee172da793a6e79d5
60e5f79bd058a12a280433ed6fa46510a
oosaru@oosaru-G7-7588:/home/saru/IFMS/2023_1/segurancadecomputadores/SegundoTrabalho$ echo "1"
1
oosaru@oosaru-G7-7588:/home/saru/IFMS/2023_1/segurancadecomputadores/SegundoTrabalho$ echo "1" | sha51
2sum
3abb6677af34ac57c0ca5828fd94f9d886c26ce59a8ce60ecf6778079423dccff1d6f19cb655805d56098e6d38a1a710dee595
23eed7511e5a9e4b8ccb3a4686 -
oosaru@oosaru-G7-7588:/home/saru/IFMS/2023_1/segurancadecomputadores/SegundoTrabalho$ echo -n "1"
1oosaru@oosaru-G7-7588:/home/saru/IFMS/2023_1/segurancadecomputadores/SegundoTrabalho$ echo -n "1" | s
ha512sum
4dff4ea340f0a823f15d3f4f01ab62eae0e5da579ccb851f8db9dfe84c58b2b37b89903a740e1ee172da793a6e79d560e5f79
bd058a12a280433ed6fa46510a -
oosaru@oosaru-G7-7588:/home/saru/IFMS/2023_1/segurancadecomputadores/SegundoTrabalho$ 
```