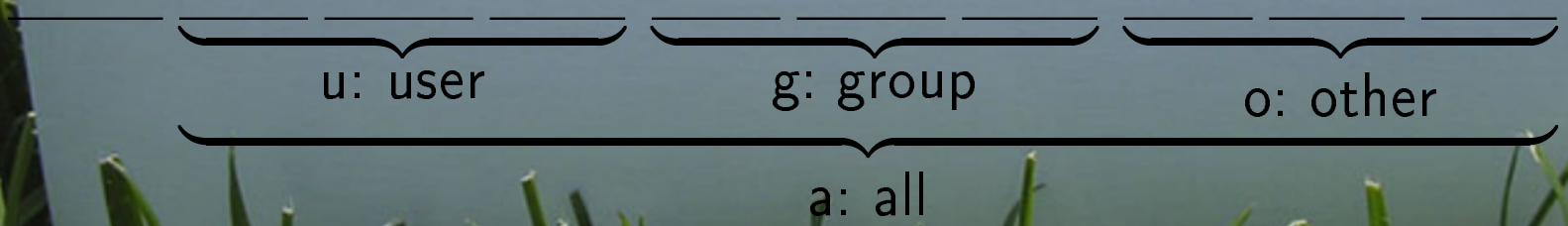
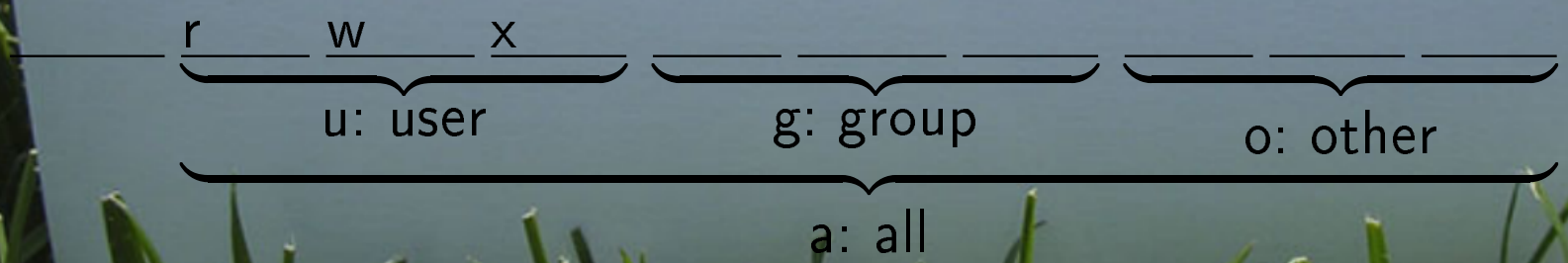
Permisos

```
    $ touch solo
    $ ls -l solo
-rw-r-r- 1 restrepo restrepo 0 2009-05-25 17:44 solo
    $ chmod a-rwx
    $ ls -l solo
```



Permisos

```
    $ touch solo
    $ ls -l solo
-rw-r-r- 1 restrepo restrepo 0 2009-05-25 17:44 solo
    $ chmod a-rwx
    $ ls -l solo
    $ chmod u+rwx solo
```

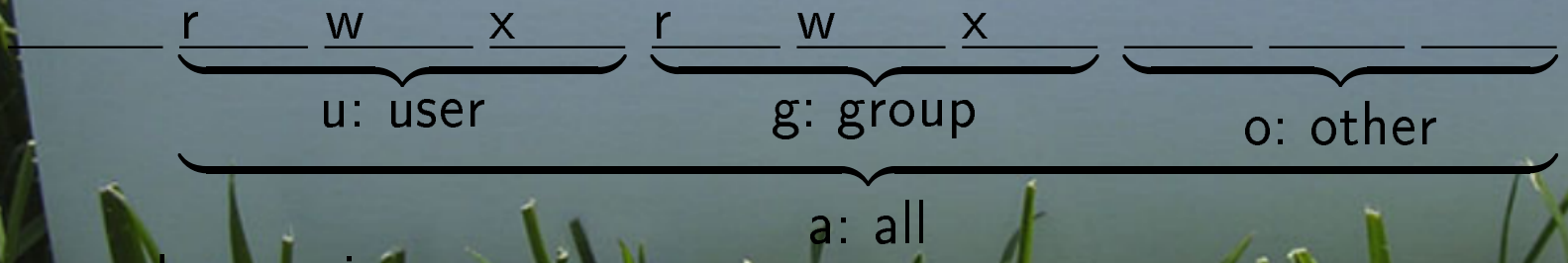


r: read, w: write, x: execute

Permisos

```
    $ touch solo
    $ ls -l solo
-rw-r-r- 1 restrepo restrepo 0 2009-05-25 17:44 solo
    $ chmod a-rwx
    $ ls -l solo

    $ chmod g+rwx solo
```

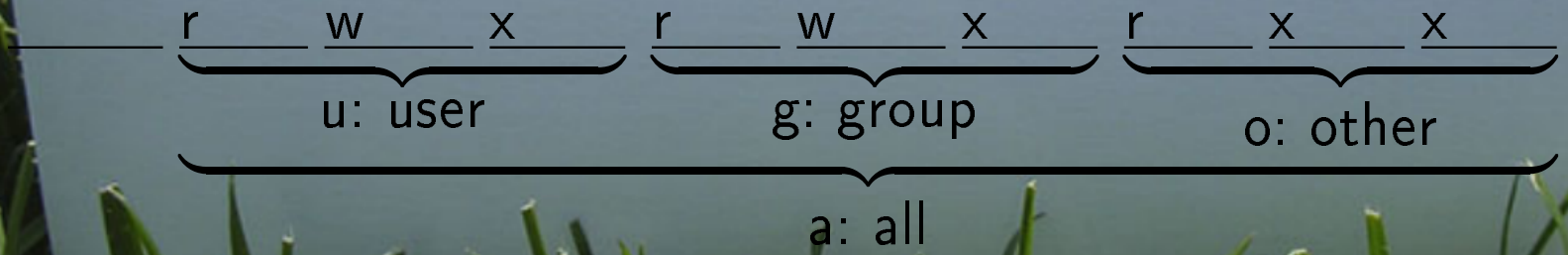


r: read, w: write, x: execute

Permisos

```
    $ touch solo
    $ ls -l solo
-rw-r-r- 1 restrepo restrepo 0 2009-05-25 17:44 solo
    $ chmod a-rwx
    $ ls -l solo


    $ chmod o+rwx solo
```



r: read, w: write, x: execute

```
#!/bin/bash
M=3 # exponente por defecto
if [ "$1" = --help ] || [ ! $1 ];then
  echo "Evalua la potencia al cubo de un entero
  USE COMO $0 [OPCIONES] N
  --help     muestra esta ayuda
  -n M       cambia el exponete a M"
  exit
fi
msg="ERROR: el argumento entrado no es entero ver $0 --help"
if [ "$1" = -n ]; then
  if [ $2 -ge 0 -o $2 -le 0 2>/dev/null ];then
     M=$2
     if [ $3 -ge 0 -o $3 -le 0 2>/dev/null ];then
N=$3
     else
        echo $msg
        exit
     fi
  else
      echo $msg
      exit
  fi
```

```
elif [ ! $2 ] && [ $1 -ge 0 -o $1 -le 0 2>/dev/null ];then
    N=$1
else
    echo "ERROR, ver $0 --help"
    exit
fi
echo $(($N**$M))
```

Terminal

```
  $ ./programa.sh -n 2 3
9
  $ ./programa.sh 3
27
  $ ./programa.sh a
ERROR, ver ./programa.sh --help
```

Tuberías

La salida de un comando se puede usar como entrada de otros por medio del simbolo “|”:

```
$ cmd1 | cmd2 | ... |cmdn
```

Ejemplo:

```
ls -l | grep ^d
```

Redirecciones

La salida de un comando se puede guardar como archivo con “>”. “2>” guarda la salida de errores de un comando. “> >” añade la salida de un comando al final de un archivo

```
$  cmd > archivo
```

Ejemplos:

```
echo "hola mundo" > solo
ls no_existe 2> /dev/null
```

Donde `/dev/null`, es el archivo nulo (agujero negro)

REGEX

Una expresión regular, REGEX, a menudo llamada también patrón, es una expresión que describe un conjunto de cadenas sin enumerar sus elementos. Es usada por muchos comandos, lenguajes de programación y editores. El patrón más conocido es “*”. El significado más común de los operadores es:

REGEX II

Operadores

()	Conjunto de caracteres a operar. Un caracter si no hay
*	Cero o más de lo anterior
.	Cualquier caracter
?	lo anterior es opcional y solo una vez
+	Uno o más de lo anterior
{N}	N veces del lo anterior
{N,}	N o más veces del lo anterior
{,M}	Hasta de M veces del lo anterior
{N,M}	Al menos N pero no más de M veces del lo anterior
^	Comienzo de línea
$	Fin de línea
[]	Conjuntos de caracteres: [a-z] conjunto de minúsculas
|	el operador or
\	El siguiente caracter adquiere un significado especial
\n	nueva línea
\␣	espacio

Ejemplos REGEX

```
$ cp /usr/share/dict/words .
$ echo 1hola2 >> words && echo abc2ef >> words
$ echo ABC123 >> words && echo 123 >> words
```

- Encuentre palabras que contengan baba babe .. bubu
  `$ grep -E '(b[aeiou]){2,2}' words`
- Encuentre palabras que contengan numeros
  `$ grep -E '[0-9]' words`
- Encuentre palabras que contengan al menos dos numeros
  `$ grep -E '[0-9].*[0-9]' words`
- Encuentra palabras que terminen en tres vocales
  `$ grep -E '[aeiou]{3,3}$' words`
- Encuentre las palabras que comiencen con x
  `$ grep -E '^(x|X)' words`, o, `grep -E '^[xX]' words`

Ejemplos REGEX

- Encuentre las palabras que comiencen y terminen con X
- Encuentre la palabra más larga
- Encuentre la palabra con más mayúsculas
- Encuentra las palabras capicuás
- liste los archivos y directorios invisibles
  `$ ls -ld .??*`
- liste los directorios invisibles
  `$ ls -ld .??* | grep ^d`
- liste los archivos del 16 al 19
- liste los archivos cuyos nombres comienzan con e, tienen números, y terminan en s

Comandos básicos

ls

cp

rm

mkdir

pwd

echo

Imprime en pantalla: `$ echo "echo hola mundo" > solo`

```
$ chmod u+x solo
$ ./solo
hola mundo
```

cat

Imprime todo el archivo en pantalla: `$ cat solo`

- La salida puede ser redirigida a otro archivo

```
cat solo > solo_backup
```

- Para copiar un dvd al disco duro manteniendo el formato de imagen

```
$ cat /dev/dvd > /home/usuario/matrix.iso
```

Y para reproducirlo se usa por ejemplo

```
$ mplayer matrix.iso
```

less

Imprime pantalla a pantalla un determinado archivo: `$ less solo`. “q” para salir. Navega entre páginas `<AvPag>` `<RePag>`.

more

Similar a less: `\$ more solo`. Descubra las diferencias

grep

Busca un texto en un conjunto de archivos

```
        $ grep hola solo
solo:echo hola mundo
```

Nombre del archivo:línea con la búsqueda

tail

head

sort

seq

wc

touch

perl -p -i -e 's/find/replace/[g]'

Encuentra y reemplaza en un determinado texto.

```
$ perl -p -i -e 's/hola/mundo/' solo
$ cat solo
echo mundo mundo
```

La instrucción opcional g realiza el cambio en toda la línea

awk [-F"SEP"] '{print $N}'

Interpreta un archivo como una tabla con las columnas separadas por SEP, e imprime en pantalla la N-sima columna. El espacio es el SEP por defecto.

```
  $ awk '{print $1}' solo #imprime la primera columna
echo
  $ awk '{print $2}' solo
hola
  $ awk '{print $3}' solo
mundo
  $ awk '{print $NF}' solo #imprime la última columna
mundo
  $ awk '{print $(NF-1)}' solo
hola
  $ cat solo | perl -p -i -e 's/ /:/g'
echo:hola:mundo
  $ cat solo | perl -p -i -e 's/ /:/g' | awk -F":" '{print $2}
hola
```

Variables

```
    $ a=echo
    $ b="hola mundo"
    $ echo $a $b
echo hola mundo
```

Sustitución de comandos

```
    $ $a $b
hola mundo
    $ c=‘$a $b‘   #asigna el resultado de "$a $b" a $c
    $ echo $c
hola mundo
```

Operaciones con enteros

expr N OPER M (Note los espacios)

OPER: + - * / %

```
    $ expr 4+4
4+4
    $ expr 4 + 4
8
    $ expr 2 \* 4
8
    $ expr 8 / 4
2
```

let OPER

donde OPER es cualquier operación de enteros con + - * ** / %

```
    $ let a=(1+2)*3/3-2
    $ echo $a
1
    $ let a++
    $ echo $a
2
    $ let "a=2 + 2"    #los espacios deben ser agrupados entre
    $ echo $a
4
```

Directamente como variable

```
    $ echo $(( (1+2)*3/3-2  ))
1
    $ a=$(( (1+2)*3/3-2  )) && echo $a
1
```

bc -l

```
s (x)  The sine of x, x is in radians.
c (x)  The cosine of x, x is in radians.
a (x)  The arctangent of x, arctangent returns radians.
l (x)  The natural logarithm of x.
e (x)  The exponential function of raising e to the valu
j (n,x) The Bessel function of integer order n of x.
```

Para obtener π

```
echo "4*a(1)" | bc -l
```

Para obtener la suma de una columna de números:

```
cat  lista_de_numeros | perl -p -i -e 's/\n/+/'| perl -p -i -e
```

Con este comando se podría hacer un programa de calculadora en línea de comandos

Comandos de sistema

du, ps, which, df, ifcongif, /proc, dd, kill