FTP storage covert channel tutorial-ish

see the last page to install vsftpd (very secure FTP daemon), an FTP server

suppose that each file on the FTP server represents a single ASCII character
- by this we mean the permissions of the file: `drwxrwxrwx`
- this works nicely because a permission "bit" can either be on (d, l, rwx) or off (-)

let's further restrict that to the basic ASCII character set (i.e., 0-127), which requires 7 bits
- the permissions of a file is made up of 10 characters
- we can effectively throw away 3 of them: `---xrwxrwx`

to add noise, we can add files that have one or more of the first three permission "bits" set
- but we ignore them on the receiving side
- problem is, this wastes space (i.e., we could use these files, plus we could use the extra 3 bits)
  - we'll deal with this later

let's try the message: **Demigod**
- first, we break it down into its ASCII representation:

```
D=68
e=101
m=109
i=105
g=103
o=111
d=100
```

- next, we convert to 7-bit binary:

```
D=1000100
e=1100101
m=1101101
i=1101001
g=1100111
o=1101111
d=1100100
```

- file permissions are made up of three categories (user, group, other), each made up of 3 bits
  - we need to prepend the bits with two 0s (to end up with 9 total bits)
  - then we split them into 3 groups (or octet)

```
D=001 000 100
e=001 100 101
m=001 101 101
i=001 101 001
g=001 100 111
o=001 101 111
d=001 100 100
```

  - next, we need to convert each octet to decimal to obtain the permission values

```
D=104
e=145
m=155
i=151
g=147
o=157
d=144
```

we can now create random files, sort them, and apply the permissions in sorted order
e.g., (in order):

```
touch file1
chmod 104 file1
touch file2
chmod 145 file2
touch file3
chmod 155 file3
touch file4
chmod 151 file4
touch file5
chmod 147 file5
touch file6
chmod 157 file6
touch file7
chmod 144 file7
```

the result is something like this:

```
---x---r-- 1 jgourd jgourd 0 Jan 3 16:15 file1*
---xr--r-x 1 jgourd jgourd 0 Jan 3 16:15 file2*
---xr-xr-x 1 jgourd jgourd 0 Jan 3 16:15 file3*
---xr-x--x 1 jgourd jgourd 0 Jan 3 16:15 file4*
---xr--rwx 1 jgourd jgourd 0 Jan 3 16:15 file5*
---xr-xrwx 1 jgourd jgourd 0 Jan 3 16:15 file6*
---xr--r-- 1 jgourd jgourd 0 Jan 3 16:15 file7*
```

adding noise means adding files with some of the first three bits set; e.g.,:

```
---x---r-- 1 jgourd jgourd 0 Jan 3 16:15 file1*
d--xrw-r-- 1 jgourd jgourd 0 Jan 3 16:15 file1.5*
---xr--r-x 1 jgourd jgourd 0 Jan 3 16:15 file2*
---xr-xr-x 1 jgourd jgourd 0 Jan 3 16:15 file3*
-r-xrwxrwx 1 jgourd jgourd 0 Jan 3 16:15 file3.5*
---xr-x--x 1 jgourd jgourd 0 Jan 3 16:15 file4*
---xr--rwx 1 jgourd jgourd 0 Jan 3 16:15 file5*
-rwx--xr-x 1 jgourd jgourd 0 Jan 3 16:15 file5.5*
---xr-xrwx 1 jgourd jgourd 0 Jan 3 16:15 file6*
-rw-r---wx 1 jgourd jgourd 0 Jan 3 16:15 file6.5*
---xr--r-- 1 jgourd jgourd 0 Jan 3 16:15 file7*
drwxr-xrw- 1 jgourd jgourd 0 Jan 3 16:15 file7.5*
```

receiving is just the reverse

```
---x---r-- 1 jgourd jgourd 0 Jan 3 16:15 file1*
0001000100=68=D
d--x---r-- 1 jgourd jgourd 0 Jan 3 16:15 file1.5*
1001000100=ignored
---xr--r-x 1 jgourd jgourd 0 Jan 3 16:15 file2*
0001100101=101=e
```

...and so on...

what about using all permission “bits” and not wasting space?

no more noise files (i.e., all files are meaningful)

let's use them all in the same manner (on or off)
10 bits per file/directory
order alphabetically, decode, and concatenate all the bits

to create the message, its bits must first be divisible by 10
if not, either add extra “fluff” characters to the message to ensure this
or append the bits with 0s and ignore those when decoding

when decoding, bits must be split up in groups of 7 (since we are using basic ASCII)
extended ASCII is not really workable at the command line
many characters are not printable
although so are characters with ASCII values 0-31...

try to decode the following:

```
d---r--rwx 2 jgourd jgourd 4K Jan 03 20:57 0fd1b45f22e18b3
-r-xrw--w- 1 jgourd jgourd  0 Jan 03 20:57 17c455d90e49
-rw--w-r-x 1 jgourd jgourd  0 Jan 03 20:57 302289542768697c
-rw---x--- 1 jgourd jgourd  0 Jan 03 20:57 4bdf419390d83b860cec
--wxr-xrwx 1 jgourd jgourd  0 Jan 03 20:57 51451ddb647ff3566601f232
d-w---xr-- 2 jgourd jgourd 4K Jan 03 20:57 6e8dd5f0924ce30b35aeaed9
d-wxrw--w- 2 jgourd jgourd 4K Jan 03 20:57 70a8cbb30
dr--r-x-w- 2 jgourd jgourd 4K Jan 03 20:57 79bf30d265cbd436079e
-rwxrwx--x 1 jgourd jgourd  0 Jan 03 20:57 81052541de641ff1ed7ca40
d-w-----wx 2 jgourd jgourd 4K Jan 03 20:57 a8b18ffb171e161c753ab8d
-rw-rwxrw- 1 jgourd jgourd  0 Jan 03 20:57 c52eda933ff95be8f914eaf62
-r-x-----x 1 jgourd jgourd  0 Jan 03 20:57 daf9509999adb4f6e6b49c7e91
d---rwxr-- 2 jgourd jgourd 4K Jan 03 20:57 f35c8e8ed0fb8a609
--wxrw--w- 1 jgourd jgourd  0 Jan 03 20:57 f4ed4ab4e61c850de968
-rwxrwx-w- 1 jgourd jgourd  0 Jan 03 20:57 f59a77545fe6d10
---x------ 1 jgourd jgourd  0 Jan 03 20:57 fce47615d2
```

solution on the next page (don't look yet!)

first file:

```
d---r--rwx 2 jgourd jgourd 4K Jan 03 20:57 0fd1b45f22e18b3
```

decodes to:

```
1000100111
```

second file:

```
-r-xrw--w- 1 jgourd jgourd  0 Jan 03 20:57 17c455d90e49
```

decodes to:

```
0101110010
```

and so on...we keep decoding

```
10001001110101110010011001010101100010000011101111010001100101
11100101100101010011111100110100000110110111110010100000110011
11000011110010011111101000010000000
```

and now to get the message (first, split into groups of 7 bits)

1000100	1110101	1100100	1100101	0101100	0100000	1110111	1101000
D	u	d	e	,	**space**	w	h
1100101	1110010	1100101	0100111	1110011	0100000	1101101	1111001
e	r	e	'	s	**space**	m	y
0100000	1100011	1100001	1110010	0111111	0100001	000000	
space	c	a	r	?	!	**ignored**	

message: Dude, where's my car?!