

**dR0**

Ruskean Reian Ritarit

**COLLABORATORS**

	<i>TITLE :</i> dR0		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Ruskean Reian Ritarit	June 9, 2022	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>dR0</b>	<b>1</b>
1.1	Droppings! Another RRR-Production! (Mar 20, 1995)	1
1.2	Packers parade	1
1.3	Introdamus	2
1.4	Requirements	2
1.5	Usage	2
1.6	Benchmarks	3
1.7	History	3
1.8	Assebler users	3
1.9	Algorhytm	4
1.10	Contact Us	4

---

## Chapter 1

# dR0

### 1.1 Droppings! Another RRR-Production! (Mar 20, 1995)

```

+-----+
|
| ..... \_   \_   \_   \_ ..... |
| ..... / /  // /  // /  // ..... |
| ..... /_   \_   \_   \_ ..... |
| `..... \/: \_   /: \_   /: \_   / ..... ' |
| ..... [ YLPEÄNÄ ESITTÄÄ TUOTOKSEN: ] ..... |
+-----+

```

(Proudly present a product called)

dR.zERo's~zero-packer

These horrors will follow you to a penthouse... and ←  
beyond!

### 1.2 Packers parade

After producing many unreleased version of their fast and compact packer, RRR's Utility Division has managed to finish it - and release it to the packer-hungry public.

«« Hallusinogendisorted dr.zeros's packer »  
© Ruskean Reian Ritarit 1994-1995

Winners don't use drugs.

1.....	
Introduction	
2.....	
Requirements	
3.....	
Usage	
4.....	
Benchmarks	

	5....
History	
	6....
Assembler~users	
	7....
Algorhytm	
	8....
Contact~us	

### 1.3 Introdamus

What is dr.zero's zero packer? It is a packer that uses RRR's inhouse packing algorhytms. They may not be very efficent, but they sure thing are fast. This baby packs just zeros, but does it really good. RRR's Utility Division has also included the assembler sources of the packing and unpacking routines. If you use them in your own program, then you must credits us.

The idea in packing zeroes is simple; lets take an example where you would like to clear a screen and draw a little icon somewhere on it. With zero-unpacker (tm) you don't have to do the clearing and the plotting. Just unpack the whole picture (which has apparently many zeroes) to the bitmap and your icon is right there where it should be.

We think it is brilliant.

And because the whole program is programmed in assembler it is fast and small in size. The Utility Division used library routines when possible and made the program 100% system friendly. You may think that is not 100% system friendly, but take a look at the code yourself.

Jolly well done.

### 1.4 Requirements

Requirements: An Amiga computer. =)

### 1.5 Usage

Usage: dR0 [filename] [enter]

That means that the packer has no options. It is very easy to use. You pack and unpack a file same way; Just feed file's name to the packer and it takes care of the rest. The packet or unpacket =) is written into the same directory where the other one lies.

---

The packer informs you in english when an error occurs.

## 1.6 Benchmarks

Here are some results

«« Procedure - Packing »»

Packer	Non 0 chrs	Elapsed time	Org. size	Packed size
dR0	0	~1s	100k	14b
Lha	0	~3s	100k	88b
Lzx	0	>1s	100k	428b
dR0	0	<3s	400k	14b
Lha	0	~6s	400k	238b
Lzx	0	~3s	400k	1472b
dR0	5	>1s	200k	44b
Lha	5	~4s	200k	260b
Lzx	5	~2s	200k	796b
dR0	100	<3s	300k	609b
Lha	100	~6s	300k	454b
Lzx	100	~3s	300k	1256b

«« Procedure - Unpacking »»

Packer	Elapsed time	Packed size	Unpacked size
dR0	>2s	14b	400k
Lha	~4s	238b	400k
Lzx	~3s	1472b	400k

Packer which were used in this competition

dR.zERo's zero-unpacker v1.0 (000)  
 Stefan BoBerg's Lha v1.50r (000)  
 Data Compression Technologies' Lzx v1.00e (020/030)

## 1.7 History

v1.0 - First public release. Uses packing routines v0.4.  
 Although the version number is 0.4 the routines are complete  
 and fully functioning.

## 1.8 Assembler users

We have included the assembly-sources of zero-packer and -unpacker here. They lie in the Asm-directory.

```
zero packer    v0.4.asm
zero unpacker v0.4.asm
```

And here are the safer version, which have the complete error detection (which makes them a little slower).

```
zero packer    v0.4.asm
zero unpacker v0.4.asm
```

If you use these sources in your own production, then you must credits us - RRR's Utility Division.

## 1.9 Algorhytm

Here is zpk's packet-format:

```
.L Packet's lenght (zpk)
.L Original lenght (bin)
.B Header (Marks the packed points)
.. Packed data
```

Zpk goes through the data and when a zero has been found it marks the spot and tries to look some more. After finding a zero, zpk counts the other zeroes and inserts the header and the number of zeroes into the packet.

Like this:

```
.B Header
.L Number of zeroes
```

Piece of cake.

## 1.10 Contact Us

```

  _/\_   _/\_   _/\_
  \___/   \___/   \___/
  |  |   /  |  |   /  |  |
  |  \_  \_  |  \_  \_  |  \_
  !__|   !__|   !__|   |
  :  1__|:  1__|:  1__|

```

E-Mail: rrr@klinja.fipnet.fi