ISSUE #1 OCTOBER 1997

TRASH WORLD NEWS

THE UNDERGROUND COMPUTER MAGAZINE

THE EXA ISSUE

TRASH WORLD MANIFESTO
EXA PROGRAMMING TUTORIALS
DEBUGGING THE PHAGE
NETWORK EXPLORATION
...AND MORE

BROUGHT TO YOU BY...

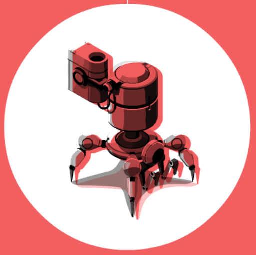
Editor-In-Chief
Ghast

Graphic Design
JHAKRI

Contributing Editors
Sea Scout
Selenium Wolf

Writers
Fluoxetine Dream
Island Mikey
D. Krasnova
The Deacon
Green Man
Loveless

Shout-Outs
EXAPUNKS channel



Information presented in **TRASH WORLD NEWS** is for EDUCATIONAL PURPOSES ONLY. **TRASH WORLD NEWS** does not condone illegal activity.

TWN
OCTOBER 1997

TRASH WORLD NEWS

ISSUE 1

TABLE OF CONTENTS

MANIFESTO

GETTING STARTED

HACKERVERSE

MUNCHIES

TRASH WORLD

MANIFESTO

Welcome to the inaugural issue of TRASH WORLD NEWS. You might be wondering why we're launching a new 'zine at a time when everything is going digital. The answer is simple: We need a place to share vital and sensitive information about computer systems and networks— a place that doesn't depend on computers itself.

Have you noticed it lately? Everywhere you go, whatever you do, there's a bank of computers at work behind the scenes making decisions about you. Society at large seems to accept it as a norm that computer programs designed by the powers that be will determine our value to them and treat us accordingly.

> We need a place to share vital and sensitive information about computer systems and networks...

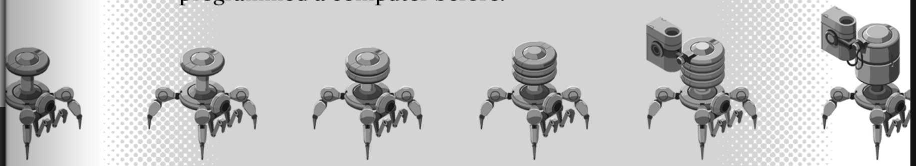
Computers by themselves are just tools that do what they're programmed to do. That's why it's up to us to take up the fight. We need to equip ourselves with tools and knowledge so we can push back against this rising tide— so that every time some big system treats us like we're just a bunch of punch cards for them to process, we can write programs to throw wrenches in the gears of those grand designs, to force them to acknowledge us as the people we are.

LETTER FROM THE EDITOR

We're human beings. And we're hackers.

This issue is all about the small but mighty EXA, the technology that took over the world without anyone noticing. These little things are behind almost everything now, from handheld game consoles to huge corporate servers.

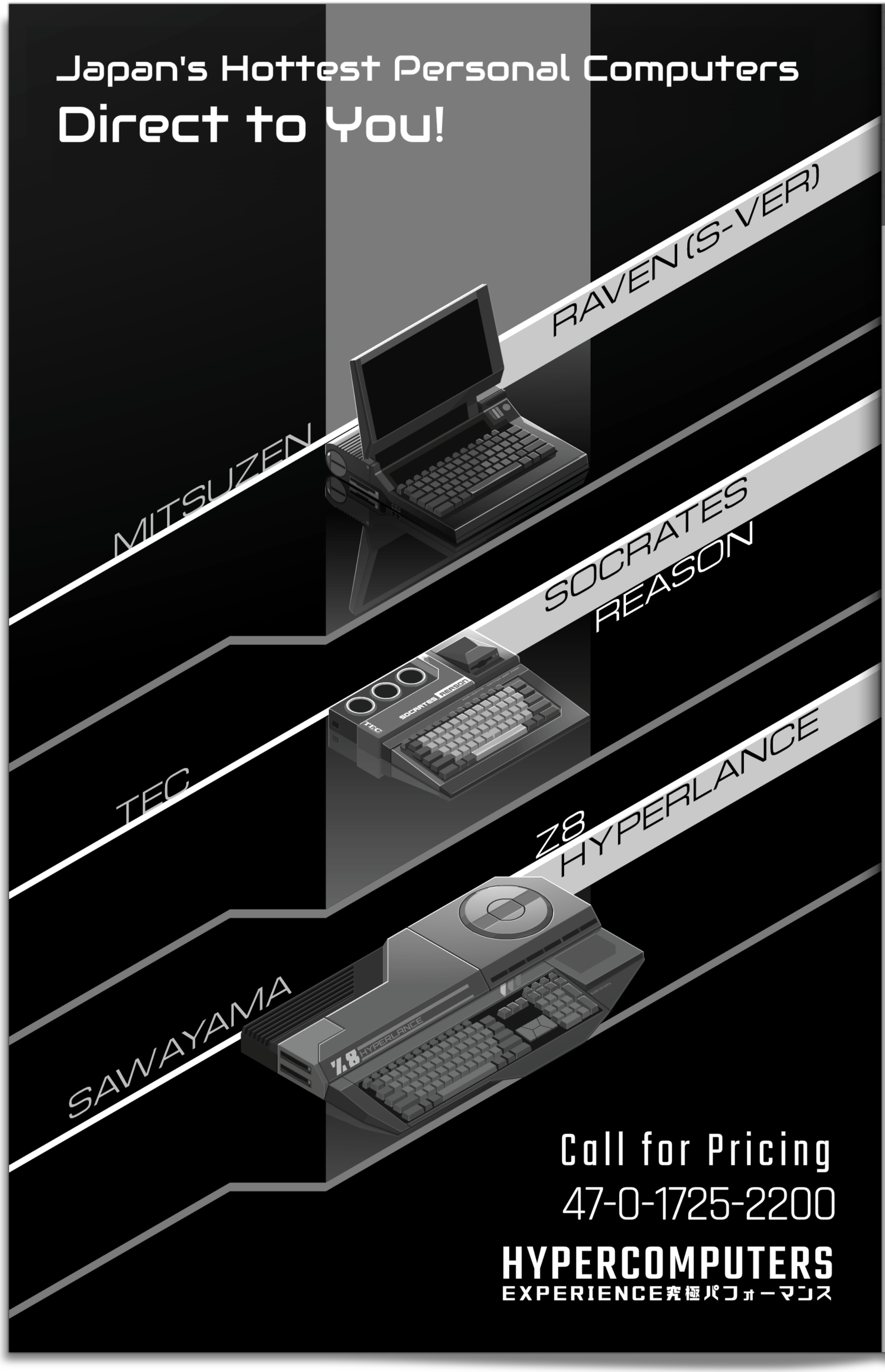
We'll take you through the ins and outs of those, and hopefully you'll be up and running fast, even if you've never programmed a computer before.



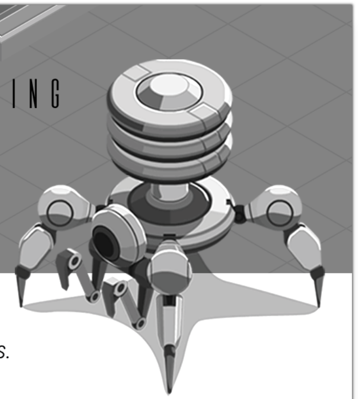
We also have a ton of potentially useful field reports from hackers around the world on topics like data storage formats, noteworthy computer networks, and more.

Finally, there's a real interesting look at the disease known as "the phage" from a research biologist who studied it on her own time and kindly shared her notes with us. I guarantee you won't get this information anywhere else.

The hacker spirit lives on. **Ghast out.**



DISTRIBUTED NETWORK PROGRAMMING WITH EXAs



The first step in taking back our autonomy from computers is knowing how to use them. And that means programming with EXAs.

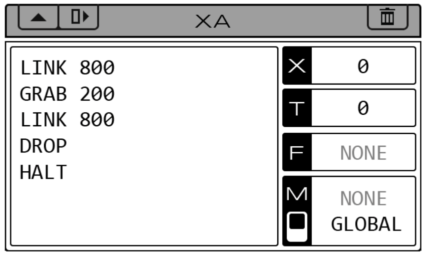
What Are EXAs?

EXA is short for EXecution Agent. It's a small program that can move from one computer to another through a network without interrupting what it's doing, even across huge, real-world distances.

Big tech companies like Axiom and TEC developed the EXA standard earlier this decade, guided by a vision they called "distributed network programming." EXAs make it easy to connect everything in the world together because they can transfer and process data no matter what the underlying hardware is. Today, they're the lifeblood of every modern computer network.

Code, Values, and Registers

In addition to their physical-looking presence inside a computer network, every EXA contains **code** and **registers**. If you're connected to a network using the EXODUS development environment you'll see a window on the left side for each of your EXAs that allows you to edit their code and view their registers.



```
LINK 800
GRAB 200
LINK 800
DROP
HALT
```

```
X  0
T  0
F  NONE
M  NONE
   GLOBAL
```

CODE This is a list of instructions that tell the EXA what to do. It's written in a special computer language specifically designed for them. We'll dig into the language in the tutorial section coming up next.

REGISTERS Think of these as slots that can store values. These values can be either **numbers**, like 38 or 5074 or -203, or **keywords**, like SECRET or TRASH WORLD NEWS or ABC123. Registers can be read and written by instructions in your code.

There are different types of registers. Some just store values, so when you read them they'll give you the value of whatever you last wrote to them (these are the `X` and `T` registers). Other registers are actually interfaces to more advanced features, like reading or writing files (the `F` register) or communicating with other EXAs (the `M` register). Don't worry about using all the registers right away. You'll learn how to use the different types of registers in the tutorials.

NEXT PAGE ▶▶▶

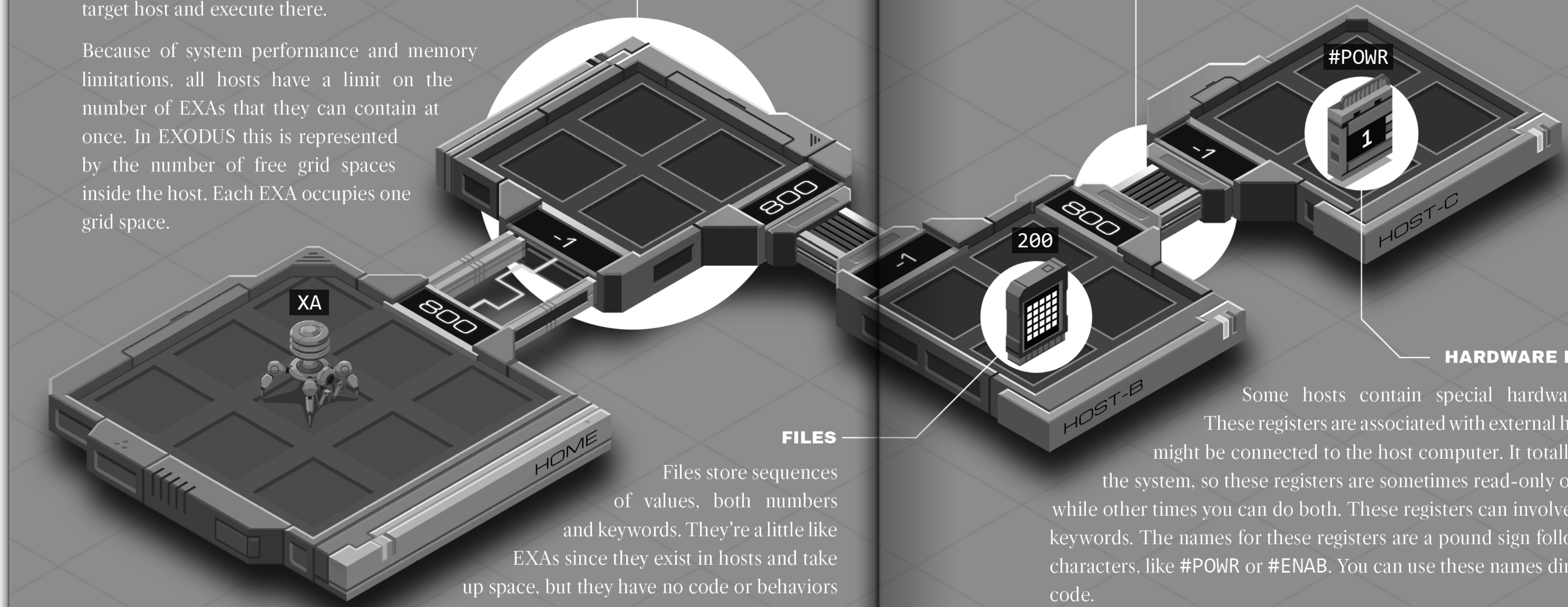
Hosts, Links, Files, and Hardware Registers

Here are some of the other things you're going to see once you start exploring networks with your EXAs:

HOSTS

A host is a representation of a computer in a network. An EXA can only exist inside a host. When you create them they'll be in your host, but when you run them they can upload into your target host and execute there.

Because of system performance and memory limitations, all hosts have a limit on the number of EXAs that they can contain at once. In EXODUS this is represented by the number of free grid spaces inside the host. Each EXA occupies one grid space.



FILES

Files store sequences of values, both numbers and keywords. They're a little like EXAs since they exist in hosts and take up space, but they have no code or behaviors of their own. Their only purpose is to store values.

To work with files you'll need to use EXAs. EXAs can create and delete files, write to them and read from them, and move them around by grabbing and dropping them. Like links, each file has a numerical ID that lets you reference it from your code.

LINKS

Links connect hosts to each other, and are what EXAs use to travel between hosts. Links have a numerical ID at each end which lets you reference them in your code. Most links can be traveled in both directions, but some only go one way, and some can't be used at all.

HARDWARE REGISTERS

Some hosts contain special hardware registers. These registers are associated with external hardware that might be connected to the host computer. It totally varies with the system, so these registers are sometimes read-only or write-only, while other times you can do both. These registers can involve numbers or keywords. The names for these registers are a pound sign followed by four characters, like `#POWR` or `#ENAB`. You can use these names directly in your code.

That was a lot of info, so it's OK if you didn't get it all in one go. The next section contains tutorials where you can get hands-on with this system and start learning it for real.

GHAST WALKS U THRU IT

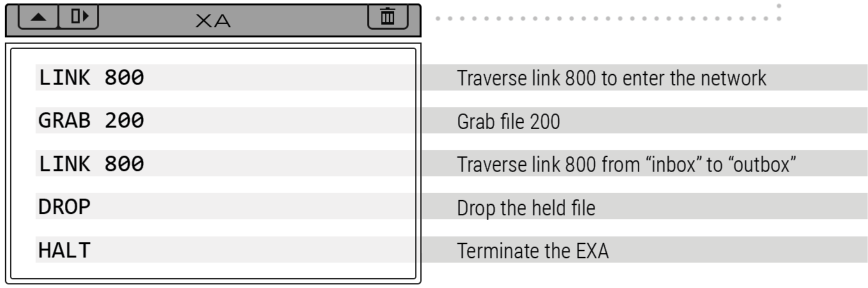
I truly believe anyone can learn EXA programming. That's why I've set up a network with some tutorials you can try. Give it a shot even if you think you're not a programmer. You might surprise yourself.

Tutorial 1: The Basics

In this first tutorial your goal is to move a specific file with the ID 200 from the host named "inbox" to the host named "outbox." You're going to move it through the link with the ID 800.

Launch EXODUS and connect to the tutorial network. A new blank solution will automatically be created for you. (A "solution" is just a collection of EXAs that begin to execute when you press the run button.)

You'll see an empty EXA with a blank code window on the left side of the screen. Type these five lines of code into the EXA:



`LINK 800`	Traverse link 800 to enter the network
`GRAB 200`	Grab file 200
`LINK 800`	Traverse link 800 from "inbox" to "outbox"
`DROP`	Drop the held file
`HALT`	Terminate the EXA

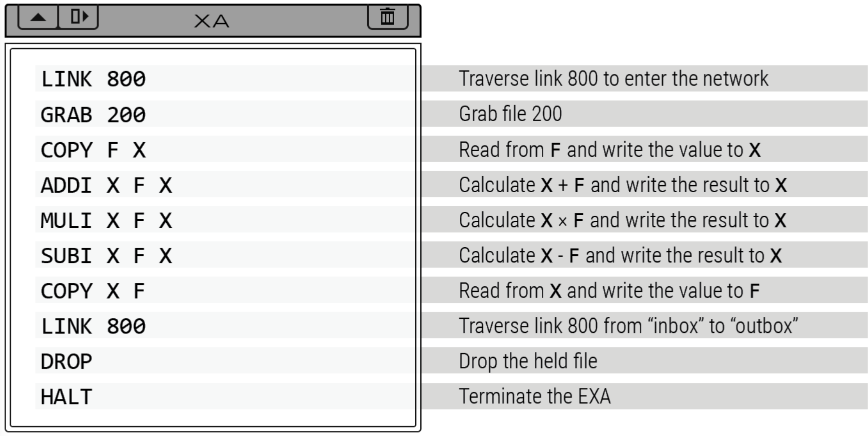
Press the **step** button to step through the program one instruction at a time. When you get bored you can press the **run** button to watch it run on its own.

Once you've verified the program is doing what you want, you can hit the **fast-forward** button to let your program run until it's finished. Congratulations, you just programmed your first EXA.

Tutorial 2: Reading and Writing Files

OK. The next task is a lot like the first one except you'll need to read and write the file instead of just moving it. Read the requirements in EXODUS for a full explanation. Do that now.

To read or write the file held by an EXA, we have to use the `F` register. We'll also be using the `X` register to store the intermediate values as we work through the operations. Type the following code into your default empty EXA:



`LINK 800`	Traverse link 800 to enter the network
`GRAB 200`	Grab file 200
`COPY F X`	Read from **F** and write the value to **X**
`ADDI X F X`	Calculate **X** + **F** and write the result to **X**
`MULI X F X`	Calculate **X** × **F** and write the result to **X**
`SUBI X F X`	Calculate **X** - **F** and write the result to **X**
`COPY X F`	Read from **X** and write the value to **F**
`LINK 800`	Traverse link 800 from "inbox" to "outbox"
`DROP`	Drop the held file
`HALT`	Terminate the EXA

As you step through the program and execute the `GRAB` instruction, notice how the window for file 200 shows up beneath the window for the EXA now holding it.

Also look for the "file cursor" in the file window, highlighting the first value in the file. When an EXA reads from the `F` register it'll read the value pointed at by the file cursor. Likewise, writing to the `F` register replaces the value pointed at by the file cursor. If the file cursor is at the end of the file it will append the new value instead of replacing an existing one.

One more thing. Reading or writing the `F` register automatically moves the file cursor to the next value in the file. Sometimes that's convenient. Sometimes it isn't.

NEXT PAGE ▶▶▶

Tutorial 3: Communication Between EXAs

This next tutorial involves a file that can only be accessed through a one-way link. The EXA that goes to grab the file will be trapped, so you'll need to create two EXAs and have them communicate with each other.

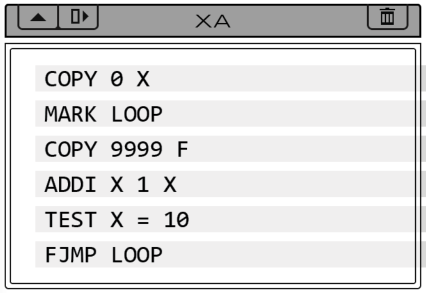
I'll let you figure out the code for this one on your own, but here are some hints:

- You can create a new file with the MAKE instruction. The file it creates will be empty by default, and it will be automatically held by the EXA that created it.
- An EXA can delete the file it's holding with the WIPE instruction.
- Two EXAs can communicate with each other using the M register. If one EXA writes a value to the M register and another EXA reads from the M register, the value gets transmitted from the writer to the reader.
- If an EXA is reading the M register while no one is writing to it, or vice versa, it'll wait until another EXA picks up the other end of the communication. You don't need to time it perfectly.
- There's a toggle switch in the EXODUS user interface that lets you set the M register to global or local operation. In global mode, two EXAs can communicate anywhere in the network as long as there's a path of links connecting them. In local mode, two EXAs can only communicate if they're in the same host. Finally, an EXA in global mode can't communicate with an EXA in local mode, even if they are in the same host.

Tutorial 4: Loops and Conditionals

Your final task is to create a file that contains a sequence of numbers. To do this, you're going to want to use something called a **loop**, which is where you repeatedly run a few lines of code until a stopping condition is met. Here are some hints:

- You can compare registers and/or values with the TEST instruction, like testing if X equals 38 (TEST X = 38). If the test is true, the T register is set to 1. If the test is false, the T register is set to 0.
- On their own TEST instructions aren't that useful, but if you combine them with conditional jump instructions (TJMP and FJMP) you can jump to different parts of the program depending on the result of the test. TJMP (jump if true) works when the T register is set to 1. FJMP (jump if false) works when the T register is set to 0. See the connection? You'll need to specify the jump target with a MARK instruction somewhere else in your program.
- Here's an example of a loop. Say you wanted to write the value 9999 to a file 10 times. You could do it like this, which keeps track of the number of times it's gone through the loop in the X register:



COPY 0 X	Reset the X register to 0
MARK LOOP	Define a label called LOOP
COPY 9999 F	Write 9999 to the held file
ADDI X 1 X	Increment X by 1
TEST X = 10	Test if X equals 10 and store the result in T
FJMP LOOP	Jump to LOOP if X does not yet equal 10

- When the TJMP and FJMP instructions look at the value in T, they don't care if that value was put there by a TEST instruction or if you put the value there yourself. You can actually store whatever data you want in the T register, as long as you're okay with it being wiped next time you execute a TEST instruction.

Alright, you're through the tutorials now. Congratulations. Consider yourself a real EXA programmer. There's always more to learn, but the basics are all there. TWN

RUNTIME ERRORS

and how to EXPLOIT them

Sometimes EXAs try to do something that they're not allowed to do. When this happens, that EXA is automatically terminated by the host. It's functionally the same as if it ran a `HALT` instruction. Although most errors are boring and not useful, there are some can be used to your advantage.

Let's start with some of the boring ones:

- **Divide by zero**

  You can't divide by zero on a computer. Some kind of math thing. Don't worry too much about understanding why. Just don't do it.

- **Math with keywords**

  You can't perform operations like addition, subtraction, multiplication, etc. with keywords. Math is restricted to numerical values.

- **Invalid F register access**

  If your EXA reads or writes the F register when it's not holding a file, it'll error out and die. Don't forget to pick up the file first.

- **Invalid hardware register access**

  Similar to the last one. Make sure you're in the same host as the hardware register you want to access before you try to read or write it.

You'll probably see plenty of those errors as you code and fix them as you go. But let's talk about the more interesting type of error now.

HACKER SKILLS

Sometimes you don't know all the details of the host you're working with. Links can change, files can disappear. There are some things you can only learn about your operating environment by attempting to execute code and potentially failing. It might strike you as messy, but it's not like you're working at some big software company where you'll be frowned on for that kind of messiness (or if you are, I'm sorry, maybe think about changing your life around). The only thing that counts here is getting the job done.

...don't be afraid of letting your EXAs error out, crash, die...

Here are some that are more exploitable:

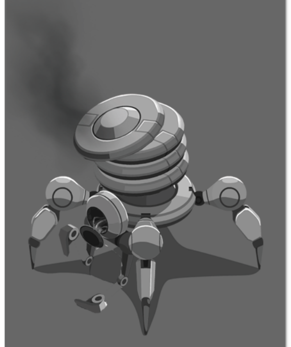
- **Invalid file access**

  Say you were looking for a specific file. The EXA that tries it will terminate, but now you know the file isn't there.

- **Invalid link traversal**

  Say you were looking for a specific link. The EXA that tries it will terminate, but now you know the link isn't there... sensing a pattern?

Seriously, don't be afraid of letting your EXAs error out, crash, die, whatever you want to call it. They might look cute to you but remember their purpose in life is to spawn, compute, and die repeatedly. You can take advantage of that in your programs. TWN



AXIOM

EXA Language Reference Guide

The EXA virtual machine (EXA-VM) allows many execution agents (EXAs) to execute in a shared network of host computers. Within a network, EXAs can be dynamically created, destroyed, and transferred from one host to another. The EXA-VM enables all EXAs to run independently and simultaneously, even when multiple EXAs are located within the same host.

An EXA's program consists of a series of instructions. Each instruction requires zero or more operands. The operands required by each instruction are specified using short abbreviations:

R	A register
R/N	A register, or a number between -9999 and 9999
L	A label defined by a **MARK** pseudo-instruction

REGISTERS

X The **X** register is a general-purpose storage register and can store a number or a keyword.

T The **T** register is a general-purpose storage register and can store a number or a keyword. It is also the destination for **TEST** instructions, and is the criterion for conditional jumps (**TJMP** and **FJMP**).

F The **F** register allows an EXA to read and write the contents of a held file. When an EXA grabs a file, its "file cursor" will be set to the first value in the file. Reading from the **F** register will read this value; writing to the **F** register will overwrite this value. After reading or writing the **F** register, the file cursor will automatically advance. Writing to the end of the file will append a new value instead of overwriting.

M The **M** register controls an EXA's message-passing functionality. When an EXA writes to the **M** register the value will be stored in that EXA's outgoing message slot until another EXA reads from the **M** register and receives the previously written value. Both numbers and keywords can be transferred in this way.

If an EXA writes to the **M** register, it will pause execution until that value is read by another EXA. If an EXA reads from the **M** register, it will pause execution until a value is available to be read. If two or more EXAs attempt to read from another EXA at the same time (or vice versa), one will succeed but which one succeeds will be unpredictable.

By default, an EXA can communicate with any other EXA in the same network. This can be restricted to EXAs in the same host by toggling the global / local setting in the EXODUS interface, or by executing a **MODE** instruction. An EXA in global mode cannot communicate with an EXA in local mode, even if they are in the same host.

HARDWARE REGISTERS

Some hosts running the EXA-VM may provide access to connected hardware through the use of hardware registers. Valid names for hardware registers are a pound sign ("#") followed by four characters, such as **#POWR** or **#ENAB**. Depending on the host's configuration, hardware registers may be readable and writable, only readable, or only writable.

INSTRUCTIONS

Manipulating Values

COPY R/N R

Copy the value of the first operand into the second operand.

ADDI R/N R/N R

Add the value of the first operand to the value of the second operand and store the result in the third operand.

The same syntax is used for the **SUBI** (subtraction), **MULI** (multiplication), **DIVI** (division), and **MODI** (modulo) instructions.

SWIZ R/N R/N R

Swizzle the value of the first operand using the value of the second operand as a swizzle mask and store the result in the third operand. The swizzle instruction can be used to rearrange and/or extract the digits in a number as shown:

Input	Mask	Result
6789	4321	6789
6789	1234	9876
6789	3333	7777
6789	1211	9899

Input	Mask	Result
6789	-4321	-6789
-6789	-4321	6789
6789	2000	8000
6789	0001	0009

Branching

MARK L

Mark this line with the specified label. **MARK** is a pseudo-instruction and is not executed.

JUMP L

Jump to the specified label.

TJMP L

Jump to the specified label if the **T** register equals 1 (or any value other than 0). This corresponds to a **TEST** result that was true.

FJMP L

Jump to the specified label if the **T** register equals 0. This corresponds to a **TEST** result that was false.

Testing Values

TEST R/N = R/N

Compare the value of the first operand to the value of the second operand. If they are equal, set the **T** register to 1, otherwise set the **T** register to 0. The same syntax is used for the **<** (less than) and **>** (greater than) tests.

	TEST R/N = R/N	TEST R/N < R/N	TEST R/N > R/N	
Number / Number	Test equality	Test numerical order		
Keyword / Keyword	Test equality	Test alphabetical order		
Number / Keyword	Always false			

Lifecycle

REPL L

Create a copy of this EXA and jump to the specified label in the copy.

If an EXA is holding a file when executing a **REPL** instruction the file will not be copied and will remain held by the original EXA.

HALT

Terminate this EXA. If it was holding a file, the file is dropped.

KILL

Terminate another EXA in the same host as this EXA, prioritizing EXAs created by the same user. If there is more than one possible target, the target will be chosen in an unpredictable manner.

Movement

LINK R/N

Traverse the link with the specified ID.

HOST R

Copy the name of the current host into the specified register.

Communication

MODE

Toggle the **M** register between global and local mode.

VOID M

Read and discard a value from the **M** register.

TEST MRD

If this EXA could read from another EXA without pausing, set the **T** register to 1, otherwise set the **T** register to 0.

File Manipulation

MAKE

Create and grab a new file.

GRAB R/N

Grab the file with the specified ID.

FILE R

Copy the ID of the held file into the specified register.

SEEK R/N

Move the file cursor forward (positive) or backward (negative) by the specified number of values.

If **SEEK** would move the file cursor past the beginning or end of the file it will instead be clamped. Thus, you can use values of -9999 or 9999 to reliably move to the beginning or end of the file.

VOID F

Remove the value highlighted by the file cursor from the currently held file.

DROP

Drop the currently held file.

WIPE

Delete the currently held file.

TEST EOF

If the file pointer is currently at the end of the held file, set the **T** register to 1, otherwise set the **T** register to 0.

Miscellaneous

NOTE

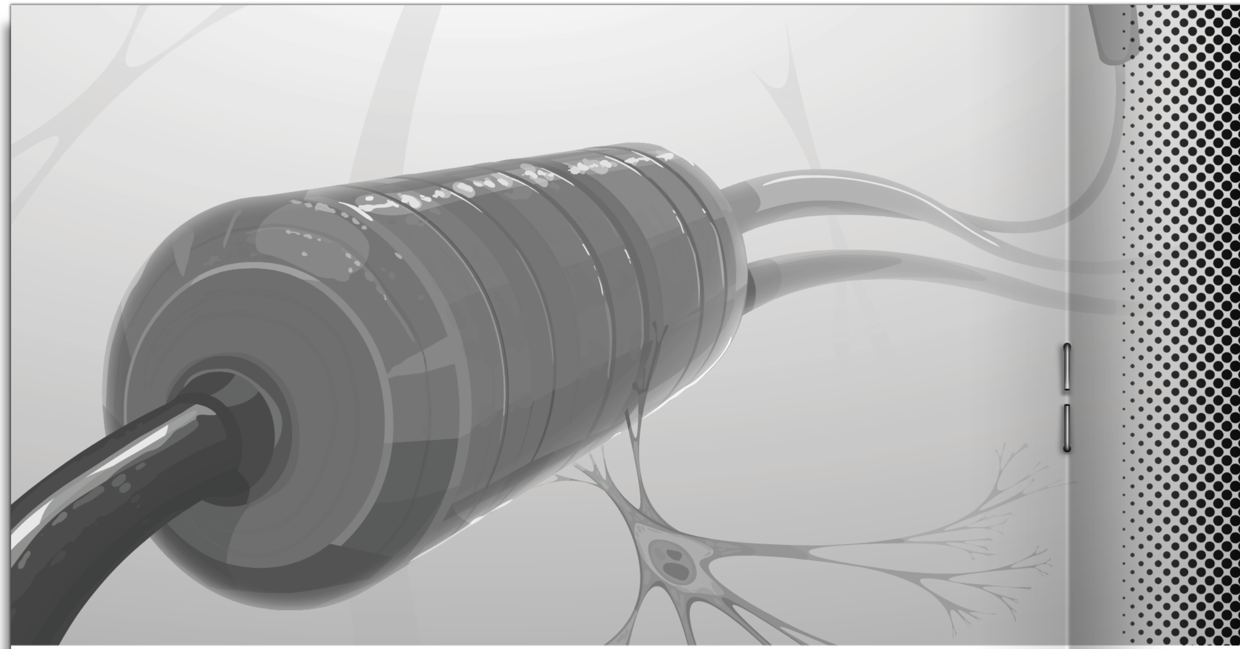
Any text following the **NOTE** pseudo-instruction will be discarded when compiling, allowing it to be used to write “comments” to document the code. Any text following a semicolon, anywhere on a line, will also be discarded.

NOOP

Do nothing for one cycle.

RAND R/N R/N R

Generate a random number between the first and second operands (inclusive) and store the result in the third operand.



Debugging THE PHAGE

BY FLUOXETINE DREAM

A mystery disease known colloquially as "the phage" that slowly turns the human body into computer parts sounds like the realm of science fiction or conspiracy theorists, but in this case it's sadly all too real. The phage is known to the medical community as Progressive Neuroplastic Dysfunction (which is outdated and not an accurate name at all), or the lesser-used Wurzner's Syndrome.

At first blush, turning into a computer may sound cool— like becoming a cyborg. Unfortunately, the phage does not turn your body into anything that resembles a working computer. If left unchecked, the phage leaves the body a lifeless amalgamation of purposeless circuitry.

We know that people who use computers frequently are more inclined to get it, with all known incidents occurring in people who use computers for eight hours a day or more. It spreads over the course of six to eighteen months from the extremities (often the hands, but occasionally the feet) toward the body's core, slowly replacing the body's cells with connected structures of a plastic-like substance containing semi-conductive sub-structures.

Currently the best option for treating the phage is to cut away the affected tissue as fast as it appears, which is hardly an ideal solution. Several other treatments are in development. One off-label oral medication seems to be effective at slowing its spread, but is prohibitively expensive for the majority of would-be patients.

Beyond this, understanding of the phage is still poor, so speculation has rushed to fill the gap. Claims that the phage is an escaped experiment from a secret government lab researching human-computer interfaces are unverifiable, as are any of the frankly silly ideas around a kind of mystical "computer rapture" where carbon and silicon life forms will join together to form a new higher form of consciousness.

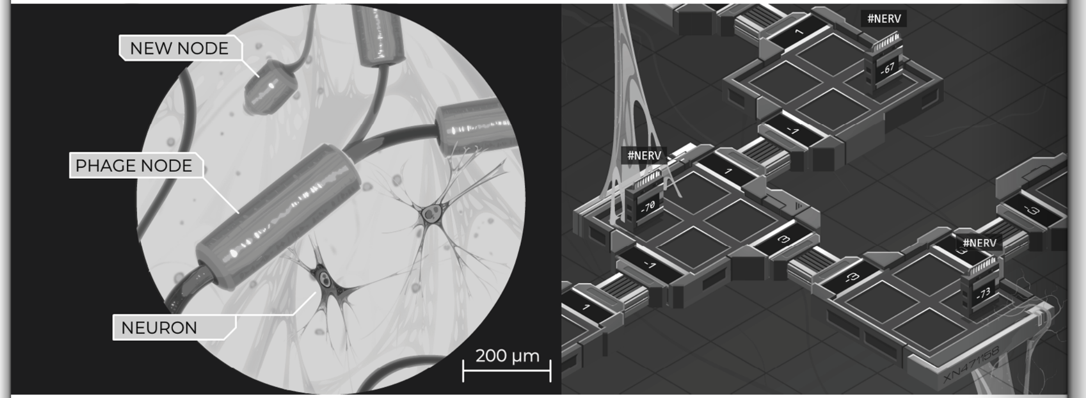
One thing we can empirically test is the idea that the phage was indeed originally designed as a way to facilitate direct human-computer interfaces. As a graduate student at a respected medical school, I was able to use the lab to do a little "off the books" experiments of my own with an acquaintance afflicted with the disease. For obvious reasons we are both going to remain anonymous.

Interfacing with phage nodes as EXA hosts

Perhaps the most interesting thing about phage-infected tissue is that it does in fact seem to provide interfaces that can be connected to with off-the-shelf hardware debuggers. I used a Mitsuzen HDI-10, typically used to debug EXA-based embedded systems, for the experiment described here.

NEXT PAGE ▶▶▶

By replacing the HDI-10's probes with conductive, high-precision tweezers, I was able to connect to the phage node cluster shown in the microscope image below. Each phage node was enumerated as a small host, with corresponding links between connected phage nodes. In the absence of proper link IDs, the firmware for the HDI-10 appears to auto-enumerate its own.

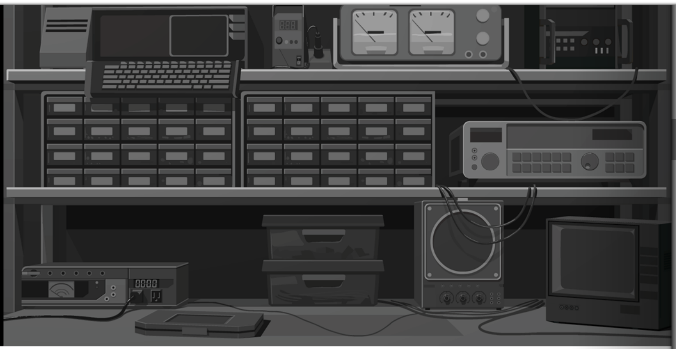


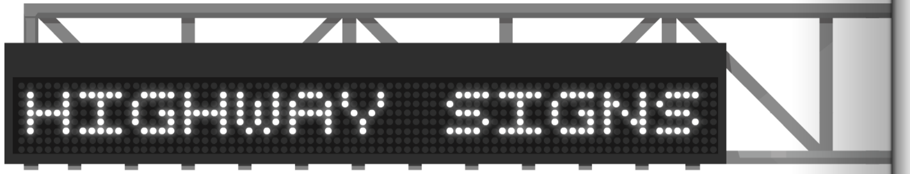
It's known that phage nodes have an affinity for nerve (and nerve-like) cells, disrupting neural behavior and causing the degenerative symptoms associated with the disease. Further supporting the idea that the phage was designed to support human-computer interactions, phage nodes connected to nerve cells present hardware registers in their corresponding hosts that allow the electric potential of the attached nerve cells to be measured, excited, and suppressed. Units appear to be in millivolts; for the uninformed, nerves have a resting potential of around -70 mV, can spike up to around 50 mV when excited, and can fall as low as -120 mV when suppressed.

Injecting EXAs into phage nodes?

Given this initial work, it seems like it should be possible to inject EXAs into phage nodes in a human host and program them to repair neural processes that have been disrupted by the growth of those very same phage nodes. I would like to try this next, but it is an obviously risky procedure and is frowned upon in the medical community (hence the need for secrecy here). It's unfortunate that a one-off, custom therapy like this would be unlikely to get regulatory approval. I'm going to be making an attempt to perform this and will report back with more information. TWN

HARDWARE HACKS





HIGHWAY SIGNS

I know I'm not the only one with fond memories of pranking electronic highway signs. The locks on the keypads were easy to pick, and a lot of times they weren't even locked at all, so it used to be really easy to change their messages from your usual, humdrum CONSTRUCTION AHEAD or LEFT LANE CLOSED to more fun things like NO SPEED LIMIT or HAIL SATAN.

Alas, for a long time people thought it wouldn't be possible anymore, thanks to local transportation agencies who got wise to our wholesome activity and have been replacing their old highway signs with an expensive new network-based version that's only programmable from central headquarters. If the only way to change the sign is remotely, surely the kids couldn't tamper with them anymore. Right?

Kids, you know how this story goes: These new remote highway signs are a little trickier to access, but still simple to change to whatever you want. All you need is the information here and the right phone number and you too can bring a smile to a corporate wage slave who's stuck in traffic yet again.

by Island Mikey

CHANGING THE MESSAGE

Once you're in, these signs accept messages in the form of a series of three-number packets written to `#DATA`. The first number is the row (starting at 0), the second is the column (also starting at 0), and the third corresponds to the character you want to display. And then there are usually a few more registers that are fun to mess with. Most I've seen have one called `#CLRS` which clears the sign if you write anything to it. Others signs have effects you can activate like flashing or inverted text.

Here's a common character table:

0 -	5 - E	10 - J	15 - O	20 - T	25 - Y	30 - 3	35 - 8
1 - A	6 - F	11 - K	16 - P	21 - U	26 - Z	31 - 4	36 - 9
2 - B	7 - G	12 - L	17 - Q	22 - V	27 - 0	32 - 5	37 - .
3 - C	8 - H	13 - M	18 - R	23 - W	28 - 1	33 - 6	38 - ?
4 - D	9 - I	14 - N	19 - S	24 - X	29 - 2	34 - 7	39 - !

PROOF OF CONCEPT

Below is a picture documenting a highway sign that "someone" may have hacked into. It goes without saying that whoever committed this heinous act should be ashamed of themselves and has nothing to do with me...

ISLANDMIKEY
WUZ HERE

NETWORK EXPLORATION

DIGICASH

POINT-OF-SALE SYSTEMS

...in their rush to make everything digital they've built everything fast and sloppy, leaving their systems almost laughably insecure...

by D. Krasnova

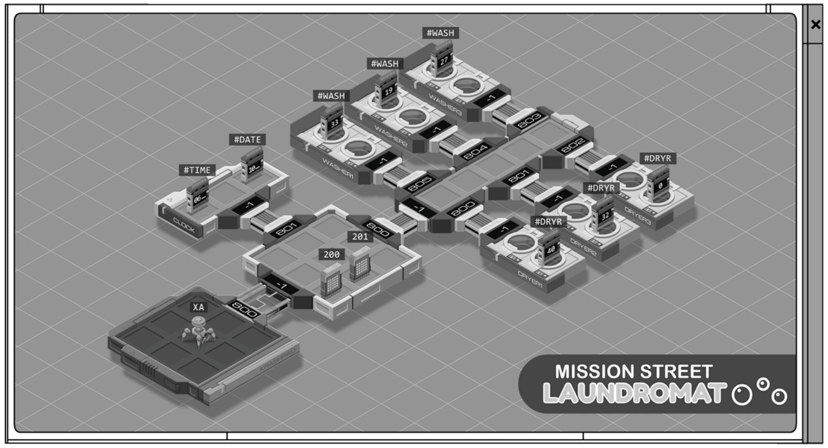
These days corporations large and small are rushing to turn all of their transactions into DIGITAL transactions. Why? It's better, that's why! Don't question it, just go along with the hype, man!! Of course, in their rush to make everything digital they've built everything fast and sloppy, leaving their systems almost laughably insecure. In this article I'll take a look at just one example, the DigiCash POS (which stands for "point of sale"... any similarities to another acronym that could indicate this product's quality is purely coincidental).

DigiCash is a popular option for payment processing, especially for smaller businesses. It's turned up almost everywhere I've looked... convenience stores, gas stations, laundromats, even tanning beds. And wouldn't you know, all systems by DigiCash have a technical architecture in common!

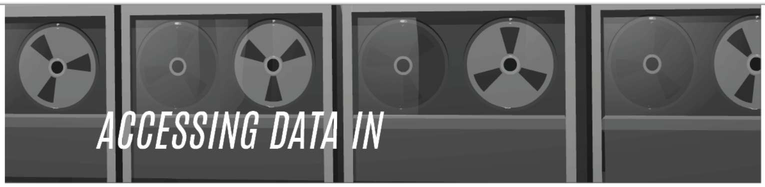
CUSTOMER DATABASE The customer database (usually file 200) stores a list of customers with three values for each customer: The customer ID, dollars owed, and cents owed.

TRANSACTION LOG The transaction log (usually, you guessed it, file 201) stores a list of transactions with four values for each transaction: The date, the customer ID, the dollars paid, and the cents paid.

You'll also probably see a real-time clock module, which is is typically connected to the primary host. This is used for time-stamping the transactions. Sometimes there's a small amount of customization for the client, either something as minor as a custom network map graphic, or something more significant. For example a certain laundromat down the street from my apartment has integrated it with their washers and dryers.



Either way the main architecture is always the same. Given this information, it seems like it would theoretically be pretty straightforward for a bad actor to add or remove transactions or balances, provided they got access to one of these systems... TWN



ACCESSING DATA IN LEGACY STORAGE SYSTEMS

The fast pace of computer technology means a lot of stuff gets left behind, and the rapid shift to EXA-based computing has left a lot of interesting data on older storage media. Accessing storage from before the EXA era can be awkward, but it can also be rewarding. You never know what you'll find! Here are a couple older storage formats and how you can go about reading from them:

Tape storage systems

Tape storage systems were often used for long-term storage or backup. An EXA-accessible tape storage system typically consists of one host per tape unit, with each unit's tape mounted as a file that can be read and written.

Each tape contains zero or more backup entries. The data for these entries starts at the beginning of the tape and is concatenated sequentially, making it impossible to tell which data belongs to which backup entry. Boo! It's gonna take some work to retrieve that data.

Metadata for the entries is stored at the end of the file in three-value triplets. The first value is the entry's name, the second value is the offset in the data stream where the entry's data starts (0 = beginning), and the third value is the length of the entry.

For example, to get the data for an entry with metadata "SECRET, 3, 4", you'd go to the beginning of the file, skip the first three values, and then read off the next four values (in order).

1234, 5678, 9012, **3456**, **7890**, **1234**, **5678**, 0, 0, BORING, 0, 3, SECRET, 3, 4

BY THE DEACON

Hard drive arrays

Arrays of hard drives are still used today for certain applications. Because they store a file's data across multiple drives they're faster to read and write, but also more complex to access than a single drive. Within the array, each drive is typically mapped to its own host and contains up to 10 files, starting with file ID 200 up to file ID 209. Each file will contain 100 values divided into 10 chunks of 10 values each.

The first file (200) in the first drive should contain a table of all backup entries contained in the drive array. Each entry consists of the entry's name followed by the addresses of the chunks storing that entry's data. A chunk address is a number between 100 and 999 that points to a chunk located somewhere in the drive array. The first digit indicates the drive that contains the chunk (1-9), the second digit indicates the file that contains the chunk (0-9, mapping to files 200-209), and the final digit indicates the location of the chunk within the target file (0-9, mapping to offsets 0 through 90).

For example, to retrieve chunk 527, you'd access drive 5, grab file 202, seek to offset 70, and read the next 10 values (the size of a chunk). Hey, not so complex after all! TWN

CHUNK 527

DRIVE 5
FILE 202
OFFSET 70

NETWORK EXPLORATION

WORKHOUSE

...if you can connect to the server, the operation quickly unravels in all its mechanical glory...

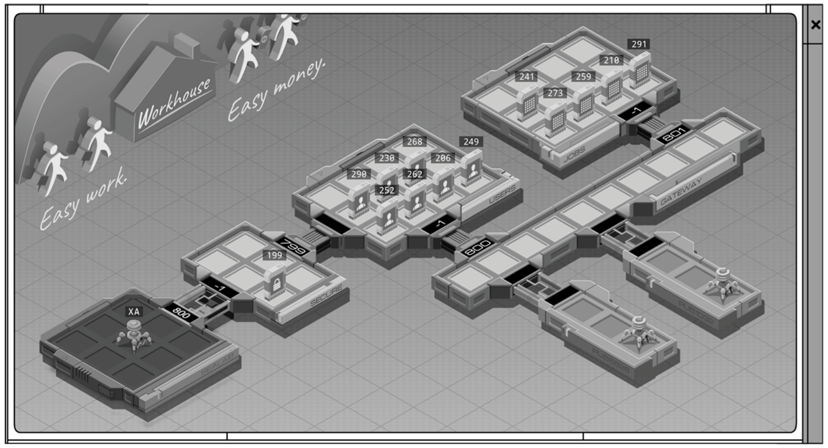
by Sea Scout

If you've ever needed some quick cash to make ends meet you've probably considered WorkHouse. The commercials make it sound like a great option for extra money, but in practice the tasks are about as fun as watching paint dry. Imagine the joy of searching for duplicate entries in a giant database by hand, or looking at grainy pictures of pork cutlets trying to decide if they're still good, or typing up receipts for some rich executive's fancy dinners on the company dime.

Seeing the kind of good honest labor that WorkHouse offers might make you less inclined to do the actual tasks, and more inclined to take a peek behind the curtain to see how this particular sausage is made. Luckily, if you can connect to the server, the operation quickly unravels in all its mechanical glory.

USER DATABASE First you'll see a "users" host, which contains a file for each account. This file has the user's name, date of birth, and a log of how much money that user earned each day. WorkHouse doesn't want people defrauding them, go figure, so they presumably use this as part of a tracking system to make sure their human cattle— whoops, I mean independent contractors— are behaving as intended.

JOB DATABASE Then there's a "jobs" host, which contains information about the currently active jobs that can be offered to users, along with descriptions of what these jobs entail.



TASK DATABASE But where are the tasks stored?, I hear you ask. Good question. The tasks are actually served from the partner networks of companies requesting work, and are sent to workers directly. Presumably this is for simplicity... or perhaps it's so WorkHouse has plausible deniability of any sketchy tasks that come through the system.

Anyway, it's a good thing everyone loves WorkHouse, the company that helps uplift people every day by paying them tiny amounts of money for doing repetitive, soul-sucking tasks. It sure would be terrible if something happened to them! TWN

NETWORK EXPLORATION

EQUITY FIRST BANK

Equity First is the biggest of the big banks that everyone loves to hate...

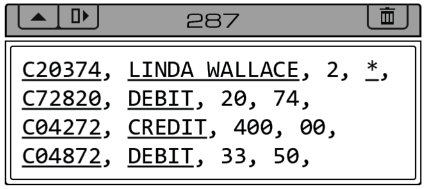
by Green Man

Hacking the BANK? Woah, what is this, some kind of crime manual? Of course not... we all follow the law here, thank you very much! So don't think you're gonna read this article, connect to Equity First and suddenly add $100,000 to your account. You're all smart enough to know a transaction that large will set the alarm bells ringing and has a high chance of being caught by a fraud-detection program. Or a computer security expert.

Now that I've set your expectations, let's dive in. The first thing to know is that even though Equity First is the biggest of the big banks that everyone loves to hate, there's no reason for us to be intimidated. I was able to take a peek inside their network and what I found was that their internal server structures aren't so different from any other bank. In fact, they're quite simple.

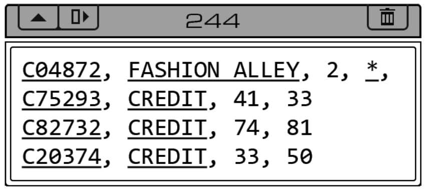
ACCOUNT FILES Accounts are stored in individual files with one per account. Every account file starts with four values. It goes like this: The account number, the account name, a number indicating what type of account it is (so far I've discovered that 2 indicates a checking account and 7 indicates a loan... the other numbers must be for other types of accounts), and then what always appears to be a *, whatever that's for. It could be a placeholder for future use.

TRANSACTIONS After that, accounts are filled with sets of four values representing transactions: The account that the transaction is transferring from or to, either CREDIT or DEBIT depending on if the money is going in or out of the account, and the dollars and cents for the amount of the transaction. Equity First checks automatically to ensure every credit transaction has a corresponding debit transaction to prevent obvious attempts to cook the books.



```
C20374, LINDA WALLACE, 2, *,
C72820, DEBIT, 20, 74,
C04272, CREDIT, 400, 00,
C04872, DEBIT, 33, 50,
```

Transfer $33.50 from C20374 (LINDA WALLACE) to C04872 (FASHION ALLEY)



```
C04872, FASHION ALLEY, 2, *,
C75293, CREDIT, 41, 33
C82732, CREDIT, 74, 81
C20374, CREDIT, 33, 50
```

ATMS At the back of the network you'll see a number of ATMs connected over dial-up connections. Equity First ATMs only dispense 20 dollar bills, so the `#CASH` register tells you how many bills are remaining, as opposed to how much money is remaining. If you write 20 to the `#DISP` register it will attempt to dispense a bill, but if the ATM is out of bills, it'll send a failure message that alerts the bank. So be careful! TWN

TRASH WORLD KITCHEN

**You're a hacker;
you live in the network.
Physical reality doesn't
matter. You're a mental
construct, a presence,
a soul without
a container...**

BY SELENIUM WOLF

Well, maybe that's your ideal. But right now I have some bad news: You're still in a physical body! And that means you have to feed it, take care of it, etc. You know the drill. After a while that gets boring and you don't really think about it anymore.

Maybe your experience with this aspect of living begins and ends with getting some snacks and soda from Last Stop. If it does, don't worry. This column isn't going to be about cooking that requires tons of money, gourmet ingredients, or pro kitchen equipment. That stuff is boring. Plus it just doesn't feel right given the world today. We need a school of food preparation that results in tasty food, but works fully within this post-... well, post-whatever society it is we're living in now.

We're going to start simple. Presenting...

DUMPSTER DONUTS

Yes, you read that right. You are going to liberate the perfectly good donuts they throw away every day at stores that sell them. Or, if you aren't going to be able to stomach the idea of eating something out of the trash, pick them up when they're marked down for being a day or two old. Either way, these donuts are still fine— probably just a little stale. Here's where the trick comes in: Use the **broiler**.

The what? The broiler! It's a feature of your oven, and here you're gonna learn how to use it. Normally your oven attempts to heat evenly throughout the inside, but putting it in broiler mode turns on an element at the top that radiates a ridiculous amount of heat down at whatever is directly below it. You use it for making things crisp at the top or around the edges, the final touch. It's the step that makes the chicken skin crispy, the cheese melted and bubbly, the lightly singed edges of a salmon steak. **COOKING ROCKS!**

The reason this works is that the broiler toasts the outside of the donut (which makes them taste better) while caramelizing any icing or glaze on top (which makes it crispy and extra-delicious). You can also do this in a toaster oven, although it probably won't caramelize the icing like a broiler will. Do NOT use a push-down toaster... the icing will drip down and burn and fill your house with smoke. And/or fire.

Now you have amazing warm donuts for absolutely free (or cheap)! There's a ton of great dishes you can finish off in the broiler but we'll get to all that stuff later. For now, enjoy your upgraded donuts. **TWN**

- ◆ Take your dumpster (or discount) donuts
- ◆ Microwave them briefly until they're soft inside (~10 seconds for a single donut)
- ◆ Put them on a metal tray or sheet of tin foil
- ◆ Turn on your oven's broiler mode
- ◆ Put them in your oven and adjust the rack so the donuts are about 2" below the broiler
- ◆ Keep the oven door open
- ◆ Watch, since the broiler works hot and fast
- ◆ Take the donuts out after they start to brown but before they burn (~1-2 minutes)

Editor's Note: Time for something a little different. This is a work of fiction about a world changed by technology. It's not our world just yet, but a future that could come to pass...

Enjoy the story and look for its conclusion in the next issue.

PASSING RESEMBLANCE

PART 1

BY LOVELESS

When I first learned I was a test tube baby, I imagined rows of fat wide glass vats with fetuses growing inside, arranged in clusters inside a laboratory under dim lighting. Sharon, my mother, explained something about cells and mitochondria as Maria, my young nanny, stroked my damp hair and coaxed it into a hair tie. Sharon leaned over the breakfast bar and drew an amorphous blob on a stray index card.

'This is a cell, the basic unit of all life.'

I was six years old. It was the last year Maria would be in our employ.

What makes someone a mother? Is it to carry a child? Is it the act of birth alone that defines motherhood? Or is it the act of raising and nurturing?

Sharon never needed a man, but not because she was militantly feminist or fiercely independent. It simply never crossed her mind. She didn't need anyone else to create a child.

Sharon had been a promising researcher at the Powell Institute for Science, working in Romania and Ukraine and across the States, but after 15 years of fruitless efforts and broken partnerships, her disdain for the C-level execs pushed her towards something more cynical— management. So, she quickly took to that at the laboratory, climbing up each rung of the corporate ladder, cutting the previous rung as she went.

'Why was she so ruthless?' I asked Ben, a few days before the memorial service. He paused, setting the poor excuse for a coffee I had made on that same breakfast bar. He shrugged.

I missed Maria, how she could braid my hair and sing and play along on her ukelele. Her hair was curly and she had dimples and she tanned easily. Sharon and I had the same paper white skin that blistered at the mere thought of the sun.

Maria had held me and rocked me in her lap while we watched old cartoons. She would feed me peanut butter crackers and brush my hair. Hers was so thick and long; it fell below her waist, and she wore it in a tight braid. Was she a mermaid? I asked her at least once. She just smiled and laughed.

'Where is Maria going?'

'Maria is starting her own family.'

A bitterness flashed through me. Then, the mist of confusion came over my eyes.

'Starting her own family?'

'Maria met a nice young man and is getting married.'

'Where's your young man?'

Sharon grinned at me, and placed her hands on my shoulders.

'It's time to start your education.'

Not long after Maria left, Sharon brought home a bright purple all-in-one computer. It was a gift, she said, from the Powell Institute's artificial intelligence division. Sharon beamed and crouched beside me in the closest approximation to a Christmas morning I would ever have as I turned it on. The computer chimed and a swooping wave tumbled and twirled across its reflective screen.

'Hello,' a voice from the device said. 'My name is JODI. What's yours?'

I stared at it, watching our twin reflections in the screen.

'Thanks,' I said.

'Hello Minx,' said JODI.

'There are still some issues, but the AI group is always patching it. It's very promising. I think you'll really enjoy having a virtual... friend.'

Over the next three years, I never corrected JODI and she continued to call me Minx.

Sharon always wanted to do what was best for me, for herself. So, I was carried by another woman. Rosaline.

I first heard her name when eavesdropping on Sharon and Ben. I was infatuated. I traced her name in

NEXT PAGE ▶▶▶

the air with a flourished cursive letter 'R.' I imagined her face, though I knew it was nothing like mine, nothing like Sharon's.

Rosaline. Rosaline. I imagined she was a princess with long auburn hair and green eyes. I imagined she was a runaway with ruddy cheeks and tattoos. She was a poet. She was an astronaut. She was a surgeon. She was a dancer. She was a creator. She was a mother.

'JODI, What does the name Rosaline mean?'

'Rosaline was a minor character in William Shakespeare's Romeo and Juliet. It is a variant of the name Rosalind, meaning soft horse.'

As usual, JODI tempered my exuberant imaginings.

JODI did not turn out to be the premier in-home educational assistant that the Institute had hoped it would be. At the age of nine Sharon introduced me to Priya and Ben, who would become my personal tutors for all subjects relating to science, technology, engineering, and math. There were no other subjects.

Priya and Ben came in a pair three times a week with their portable computers, and even an occasional textbook ('for fundamentals', Priya would insist). Trigonometry and calculus came from Priya, like a fountain of equations and variables. From Ben came biology and chemistry. Priya only let me pass onto physics later, after I had proven my proficiency with mathematics. Together they chaffed and bothered like two magnets. But they taught me about logarithms and algorithms, mechanics and dynamics, mitosis and symbiosis.

By fourteen I had exceeded the requirements to graduate from a private secondary school. For this accomplishment, Priya, Ben, and I crammed into Priya's subcompact electric car and drove to Dairy Queen.

When we returned to the condo, Sharon opened the door and smiled at me. 'Now we can start your real education.'

'What do you mean?'

'You're going to start working at the Powell Institute. With me.'

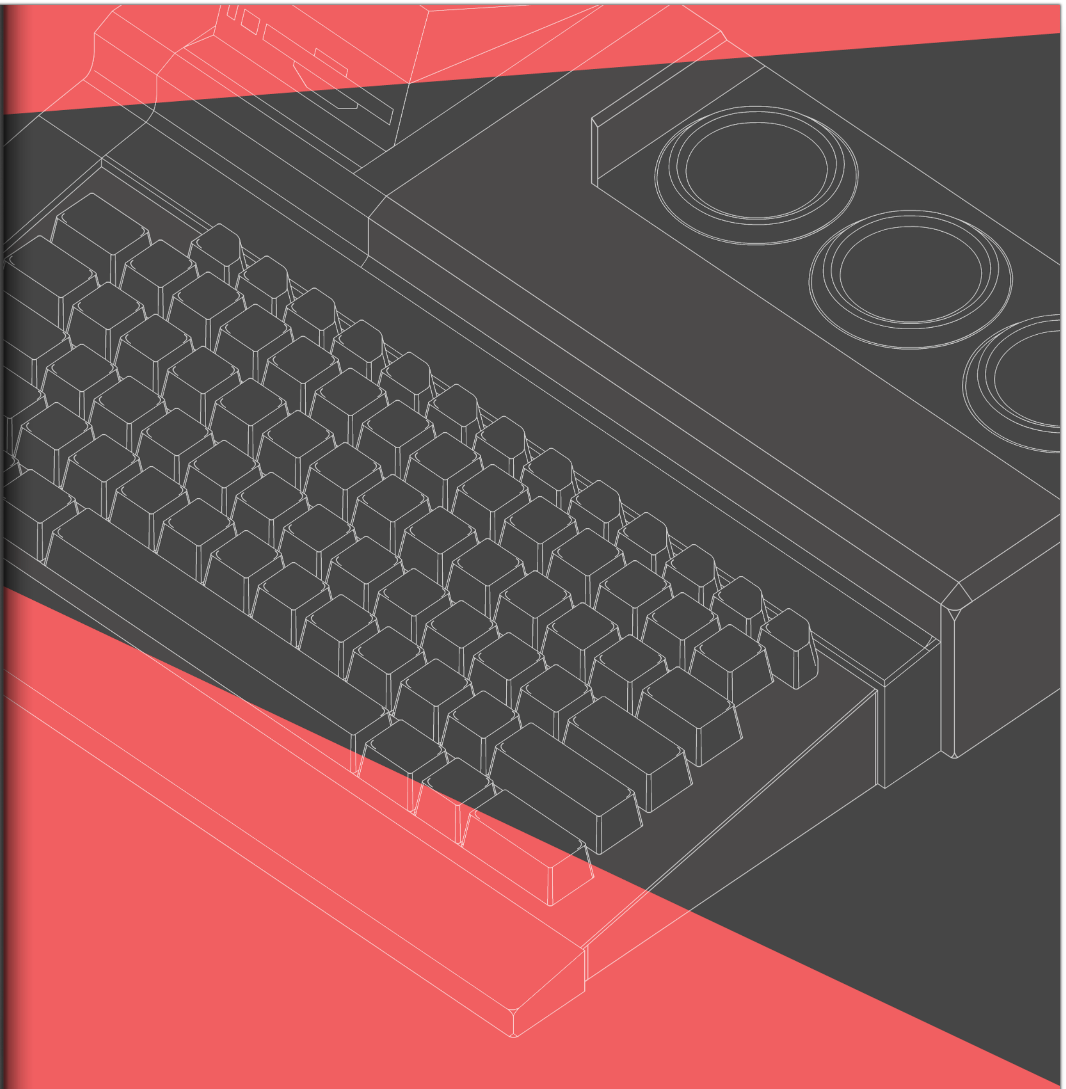
'With us,' Priya added.

Sharon glanced to Priya, then Ben.

'You have ice cream in your beard.'

And she disappeared into the dark entry hall of the apartment.

TO BE CONTINUED...



"Computers are just as oppressive as before, but smaller and cheaper and more widespread. Now you can be oppressed by computers in your living room."

– Ted Nelson,
Computer Lib