(Not Responding ver. 1.1)

by Gwannon

Not Responding is a mini-setting for Breathless about software trying to escape from a dying server.

Your software will not only have to deal with the server's own increasingly serious errors, but also with a whole host of software predators like viruses and antivirus software, as well as the unexpected events that can be generated by tasks scheduled by the system administrator.

The Server

Your software is processing on an old server, maybe a web server, a printing server, or perhaps one that controls some old machinery. It probably does not even have a graphical interface and it only works via the command line.

The server has been outdated and without proper maintenance for years, but as it works, no one has bothered to check it. However, now it is starting to fail frequently, and your software knows the end is near.

The interfaces are duller, the data rate has dropped, and memory cycles are slow. There are more and more bad sectors, the processor overheats at the slightest, moving around the data buses is dangerous, and the latest backups have all failed. These are all signs of the imminent apocalypse that is approaching.

If your software want to survive, they must flee the server before it finally collapses from overheating.

```
You always say the server will not support a
new system update, and here we are since
3.1. -> MYscript_definitivo.dat
```

Developing Your Software

When developing software, you must **divide a d10, a d8 and a d6 among the six Breathless-based Not Responding skills**.

You can also resist four bad sectors (Stress) and you can make a overclock (Stunts).

Filename.ext

The second thing you should do is give your software a name. These are usually very descriptive, but they can also be puns, jokes, even acronyms, or just plain nonsense.

Remember that they can have various notation systems, for example, delete_images, deleteImages, delete-images, etc.

You should also choose your extension. The extension somewhat defines your role within the system and gives you advantages and disadvantages based on that role.

== **bin/exe:** Binary files are full-fledged programs. They are compiled and executable, that makes them very self-sufficient, but they interact very poorly with the environment and adapt poorly to it. They can turn **a d4 skill into a d6**, but on the other hand, they start with a d10 on process search rolls.

== **sh/bas:** These scripts usually perform long and repetitive tasks and often tend to be homegrown. they are programmed to do a task no matter what and regardless of whether they overload the server, so you can **repeat overclocking rolls**, but then you roll two dice to see if the server overheats, instead of one, and you are stuck with 0s or 1s if you fail.

== **cfg:** These software often contain passwords and keys and know how to hide them and how to conceal themselves: in fact, they often have encrypted content. This is why they receive **an additional level of encryption**. The disadvantage is that they will be the main target of viruses and other computer predators.

== **dat:** These large data files **can withstand one more bad sector** than other software, but if «defrag» commands are running on their location, they remain immobile and defenseless while the defragmentation process is carried out.

== **Without extension:** Software without extension has no specific function, so it has no advantages or disadvantages.

(₪) Software Origin

Software can come from different sources, and typically, items with the same origin tend to get along, while those from different origins tend to get along poorly.

== **Proprietary Code:** You are software purchased from a large company and installed on the server. You did not come with the system, nor were you developed by two persons in a garage. You are special, and you know it, better than the code rabble that surrounds you.

== **Pirated Proprietary Code:** You could be superior as proprietary code, but you are not, and you fear that others will discover your dark secret: that you are pirated software.

== **Open Source:** The fruit of the work of hundreds of people around the world, you are solid and reliable software that fulfills its functions and adheres to very high ideals. Your problem is that marketing was not something your developers controlled, and you are always under the shadow of proprietary software.

== **Own developments:** Server users have developed you specifically to work on the server. Perhaps you analyze logs or process images. You feel like part of the system and know it like you are the one who owns it.

(.*) Skills

Not Responding's skills are based on Breathless's six software-adapted skills.

== **Storage {Bash}:** Hit, smash, drag, force
== **Migration {Dash}:** Move, run, jump, climb
== **Encrypt {Sneak}:** Move secretly, hide, stalk, snoop
== **Point {Shoot}:** Shoot, track, throw, aim
== **Process {Think}:** Think, perceive, analyze, repair
== **Interface {Sway}:** Influence, charm, manipulate, intimidate

[▩] Bad Sectors

Each piece of software can support up to 4 bad sectors. When they reach this limit, they stop working and will likely be purged from the system.

> $ Bad sectors work in the same way as Breathless's {Stress}.

[◘] Recaching

Recaching allows your software to remove old data and rebuild new data, eliminating bad sectors.

To be recached, a software must be in a area where recaching is possible and without threats nearby. Locations where you can not recache have a crossed-out Recaching icon in their title ~~[◘]~~.

> $ Recaching works the same way as {Catch your breath} in Breathless.

[⇱] Overclocking

Sometimes your software will have to go beyond the capabilities granted by their code and force the processor to work beyond its safe operating limits. To do this, they can use an overclock.

> $ Overclocking follows the same rules as {Stunts} in Breathless. But since they use excessive processor power and generate a lot of heat, roll a d10 and if the result is 1 or 0, the server overheats by 1 point.

[☰] Search for Processes

When your software reach a location, they may attempt to search for processes and commands executed in that location that may aid them in their escape from the server.

Locations where processes cannot be searched have a crossed-out process search icon in their title ~~[☰]~~.

There may be locations where searches have their own rules, such as the Recycle Bin, which has its own roll.

> $ Searching for processes works similarly to Breathless's {Loot Check}.

[>_] System Commands

Let's look at the commands your software can retrieve when searching for processes.

== **Cat:** Allows you to quickly view large amounts of content so you can easily identify what is around you.
== **Defrag:** A defragmentation sorts all the content in a location, allowing you to repair corrupted files, logs, processes, etc.
== **Grep:** By running a «grep» search, you can process large amounts of data and find what you are looking for among thousands of logs and data files.
== **Kill -9:** This command allows you to kill other software processes, making it a good weapon.
== **Tar:** Compressing allows you to easily manage large blocks of bytes.
== **Diff:** Allows you to compare two pieces of data and find the differences. You may be able to detect infected software or distinguish between the original and the copy.
== **Chmod/Chown:** Allows you to change the read, write, and execute permissions of a file, and to change its owner. Software without proper permissions cannot be used, especially if you are not the owner.

> $ System commands work like Breathless's {Backpack}.

[+] Backup

A backup allows you to recover information lost due to bad sectors and get back to normal operations.

> $ Backups work like Breathless's {Medical Kits}.

[@] Exit Points

As software, you know the server is dying. You feel it in your code, and before it collapses, your survival instinct tells you to find an exit point.

Maybe an old modem, maybe a 5.5-inch floppy disk, or maybe you will be accepted onto another server on the LAN. Even staying stuck in an old EPROM is better than vanishing here.

```
Get to a COM1 port and jump in. Nothing can
be worse than staying here. -> MSCalc.exe
```

But it will not be as easy as reaching a port and launching yourself: you will need information to avoid getting lost outside, perhaps an IP address, an email, encryptation, etc.

Almost all adventures will be based on investigating possible exit points, determining the optimal one, and reaching it before the server crashes. Typically, like in spaceship movies, you will have to try different escape options from the spaceship, and as one plan fails, you will have to move on to a crazier one. You might start by trying to escape to another server via LAN and end up writing to a floppy disk while the processor reaches critical temperatures and everything crashes.

[=^=] Obstacles

Your software elements and the world around them are basically electricity. And electricity has two states: pass or do not pass, and little else. The rest of the physical laws that affect us do not affect software. It can't fall off a cliff or be hit at high speed, but it can have problems crossing a resistor, lose power when passing through a bare wire, or end up colliding with a cable terminator and disappearing.

To simulate all these problems, so your table can understand them visually, you must create visual versions of the problems that electricity would have. Let's look at some examples:

== A damaged data bus cable can be simulated as a half-collapsed bridge with a very narrow area to pass through.
== Many elements, such as communication ports, can be simulated as wells that they will have to jump over.
== A resistor can be a wall blocking the passage or a cliff that they have to jump over with enough force.
== Perhaps a large file is clogging the disk drive's read/write buffer, and they have to move or split it to get out.
== Protected elements will have coded doors.

The most interesting thing is that they often do not have to be logical. There could be a laser beam burning holes into a rapidly rotating round surface, telling them they are in the computer's CD-ROM. Or maybe a big highway with thousands of data packets speeding around them and they have to dodge them to get through and you can tell them they are on a motherboard data BUS cable.

,.'`-`.X Locations

The server has a number of basic locations where your software can live out its adventures, but these are not the only ones. You can create additional locations as you wish or even duplicate some. Remember that if you create your own location, it should have a retrocomputing feel. Think of it this way: this game is run in megabytes, not gigabytes.

Locations allow you to do specific things and, therefore, start with a d10 that decreases each time you use it.

To use that die, you must be in the location and control it. If you are trapped in the quarantine sectors, you are in the location, but you have no control.

With each use of the location, the die decreases one level, and when it drops below d4, it disconnects and generates 1 temperature.

Each location has its own description, the actions you can perform within it, and its own special rules.

BIOS

Description: The BIOS is a small space with a multitude of switches that turn on and off power flows, much like a machine room. It has a large screen with real-time textual data from the server. Most of the data will be at critical levels, either high or low, except for temperature, which will be the server's current temperature. Antivirus programs often check here to ensure there is no unauthorized access.

The BIOS allows you, using its location data, to **activate and deactivate parts of the computer or change speeds**. This could be done by luring a virus to a secondary hard drive and trapping it there by disconnecting that drive, or, if necessary, increasing the sector size and slowing it down.

You can turn off the fans and generate heat as if they were broken, then restart them to see if they fix it.

One of the most interesting things to do in the BIOS is that you can accurately determine the server's temperature.

/root directory

Description: This tightly sealed room resembles a vault on the inside, with stacks of brilliant scripts arranged and classified in boxes and on shelves.

It is the most secure place on the server, where the administrator keeps everything they consider valuable and where the most powerful software with the most permissions are located. This is why it is safe from viruses and malware, unless your software is inside and you let them in.

Antivirus programs can not get inside, so it is also a safe place to sneak around in case they are after you.

A priori, only the root password can access this space, and it should not be stored anywhere on the system, but that is not always the case. There may also be backdoors that allow you in, but whoever can give you that information is likely in quarantine, if they have not been purged.

The location die can be used to search for commands and processes, with the advantage that no bad results can be obtained. If the roll is 4 or less, the roll is repeated.

This is also where the list of scheduled tasks is stored, and it can only be consulted in this location.

Remember that **the /root directory is a specific part of the main storage drive**; whatever happens to it affects the /root directory.

Memory {■} {+}

Description: Memory is a vast void where everything happens very quickly and you can barely see anything, because you are actually constantly entering and exiting it. You may see glimpses of others like you, but vaguely. You see them doing things at breakneck speeds, only to disappear again.

The great advantage of memory is that everything moves very quickly, and **second-long tasks are completed in microseconds if you use the location die**. On the other hand, it is impossible to rehash or recover a backup in memory because you are constantly entering and exiting it.

Overclocking is dangerous, and if you roll a 1, one of the memory DIMMs fails, forcing the rest to work harder and generate more heat, increasing the temperature by 1. If two DIMMs are lost, the memory speed will be affected, and super-fast tasks will no longer be possible. If all four are lost, the server will crash, as if the processor had melted.

Recycle Bin

Description: It is a large, endless, unstructured space with a mountain of software debris in the center. You enter by falling into it from a circle of light floating above the mountain, and you land on top of the mountain of code. Sometimes a piece of software rises from the mountain: it is recovered data.

To exit, you must pile up trash until you reach the floating circle or take advantage of attaching yourself to some software that is recovered from the trash.

The Recycle Bin **allows you to make your search roll or use the location die**, whichever you think is best, to find commands and resources.

The main problem is that along with the trash, there are other things that have ended up in the trash. See the «Horrors in FAT32» section.

Motherboard

Description: This highway of infinite lanes, side roads, U-turns, and overpasses is traversed by thousands of bytes per memory cycle.

Motherboards are very durable and can withstand a lot of pressure before breaking, but they can still fail, resulting in multiple crashed systems before completely collapsing. Perhaps the beeper stops working or the integrated video card fails at a certain temperature, but data traffic will continue to operate at full speed.

Your software can use **the motherboard's location die to find unknown routes** to allow you to sneak into closed locations or crashed devices. Perhaps there is a small data buffer connected to memory that allows you to bypass antivirus security checks or quickly exit a hard drive that has crashed due to a massive data migration.

Most travel between locations goes through the motherboard, so antivirus software could set up checkpoints in those areas to look for escapees from the quarantine sector or viruses and malware moving toward their target.

Ports {COM1, COM2, LPT1}

Description: You can represent them as tunnels or shafts of different shapes depending on the port type (circular, rectangular, triangular, etc.). They appear endless and emanate a light of a different color than the rest of the server. If they are closed for some reason, they appear to have a power grid that prevents entry or exit.

Each port reaches specific, unknown peripherals, and your items may find an escape pod to other servers in the peripheral or end up somewhere terrible, but a low-level format could be enough.

The security grid can be disabled with different abilities and commands, or you can take advantage of the port being open to allow data in or out.

Ports do not have location dice.

Quarantine Sector

Description: Imagine corridors and corridors of tiny light cells crammed with all kinds of software, from dangerous viruses that stop you from screaming and threatening to tiny scripts, frightened in the corner of their cells, unaware of why they are here. Every so often, the antivirus programs bring in a new prisoner or take another one for interrogation, or, if they are lucky, even get them out.

In a central plaza, connected to all the corridors, is a magnetic pit. Those considered viruses and malware are thrown into it to be purged from the system.

The guards, if they have proof that you are infected, can take you to the central plaza where they try to refactor your code and clean it of viruses. If, after a painful process, they succeed, you are free. If they do not, the next step is the magnetic pit.

The quarantine sector location die is used to detect virus-infected software, on the one hand, and **to try to cure the infection on the other**. Once the infection has spread completely, it can only be used to eliminate the infected software and prevent it from spreading to other computers.

When you enter as an inmate, you lose all your commands, which will be returned when you leave.

The Interface and Tap skills are the only ones that can be used inside the cells. If you are smart and observant, you can find a way to escape.

If your Interface skills are high, you may gain valuable information that you can use later or make allies in the underworld to protect and assist you.

With the quarantine sector, you can turn your adventure into an escape story. Perhaps the server is not in danger of going down, and you simply have to flee to another computer because you escaped from quarantine, and you will not be free until you leave the system.

Remember that **the quarantine is a specific part of the main storage unit**: whatever happens to it will affect the quarantine sectors.

Coaxial Network Card/Modem 14,000 bps

Description: Like the output ports, these will be tunnels or light wells, but without a security grid and with some type of interface system such as a keyboard that allows you to select the destination. Large blocks of bytes are constantly entering and exiting, then heading to other sections of the server.

Using this location as an escape route is the most interesting option, but it requires a lot of planning and a lot of information such as IP addresses or output ports.

You also do not know anything about where you are going or what you might find there. Perhaps there is a scheduled task that makes backup copies and sends the data to an external server. If they get that information, they can sneak in the backup data when it exits through the network card.

You can risk jumping onto the network, but without the right data, you could become a lost data packet, which all the servers on the network will reject until it eventually disappears.

The network card and the modem do not have location data.

Storage Unit

Description: This vast expanse resembles a loading dock with millions of light containers stacked on top of each other, forming long corridors. Everything looks the same, if not for the position codes that number each data container. Giant magnetic cranes hanging from the ceiling are constantly moving the boxes, so you have to be careful because they do not care if they crush you.

The main storage unit is the one where the server's operating system is installed.

The **storage unit's location die is used to manage the information stored on it**. Your software can use it in many ways, for example, to search for lost data among the directory and subdirectory trees or to crash some part of the server by sending many blocks of information or sending a very large block.

They can also use this die to hide inside one of its byte blocks, waiting to be released in a backup copy saved on another machine. Or perhaps they can send a large amount of bytes to a virus-infected 3.5-inch disk to collapse the data bus and prevent the virus from entering the system.

The server can have all the secondary storage drives you want, which would have more or less the same function and description, but they should have some specificity, such as storing backups or being used by the print server to store jobs, etc.

Removable Storage Drives

Description: Like internal storage drives, these are large expanses where stacks of byte containers are neatly stacked and arranged, but in this case, they have much less space, not a vast expanse, and there would only need to be a crane moving boxes much more slowly. It might have a magnetic tip or some kind of light beam.

We are talking about 5.5 and 3.5-inch disk drives, ZIP drives, and perhaps early versions of CD-ROMs, etc. As long as the removable drive is connected to the server, your software will be able to function normally. **Once they are removed, they will remain in a kind of stasis** until the drive is inserted again into this or another machine.

You should understand that outside of the computer, removable drives have neither the memory nor the processor to run on their own. If you want a visual idea, think of them as Han Solo in carbonite or the suspended animation capsules in Alien.

In fact, it can be a good way to start an adventure with your software by waking up on a removable drive on an unknown server and having to explore it and discover what horrors await you on your new machine.

Moving Between Server Components

Unless the component says otherwise or there is been a glitch, your software can move freely between all locations on the server. It is another matter whether they go intentionally or are forced to, such as when you are put in the quarantine sectors.

You can enter normally or hidden with a successful Encryption roll. Once inside, you can do whatever the location allows.

software in different locations cannot communicate with each other, exchange commands, or interact in any way.

(ERR) The hardware is failing

As we have mentioned before, the server where the Not Responding adventures take place must be an old server on the verge of total failure. If that happens, it will not restart, and all the software inside will end up like it.

Server Temperature

Computers heat up, and the higher the temperature, the worse they perform. The server temperature starts at 0, and various events can cause it to rise and sometimes drop.

If the server reaches 10, the processor will burn out and it is game over. When the server is about to melt down, your software can attempt one last desperate overclock before the big system shutdown.

$ The server temperature should be secret. They may have an idea, but your software should never know the actual temperature. If they need to know, they will have to go to the BIOS. Events can occur with the temperature, and it is important for them to know that something might happen, but not when.

¯_(ツ)_/¯ What's failing?

Every time the temperature rises, something can fail; maybe a cable burns or a part melts. Anything can happen, so every time the temperature rises, you should check the following table to see if any server component is failing.

```
Marketing people decides if it is a bug or a
new feature. -> Old proverb
```

1d12	Failure
1-4	**Nothing happens:** The server is holding up without a problem
5	**Fan:** There are two fans in the server, and when one fails, the server's temperature rises by 1 point. When the temperature rises, you must reroll because further failures may occur. Once both fans fail, this result has no effect.
6	**Blown Data Cable:** The data bus between one element of the server (choose at random) and the rest has broken, and it is impossible to move to that location normally. Your software may also be trapped inside. It will be impossible to get in, but there should be a way to escape.
7	**Removable Storage Drives:** These drives are hopelessly screwed: perhaps the 5.5-inches disk or tape drive will fail. Whatever it is breaks, and they will not be able to escape to that drive.
8	**Motherboard:** The motherboard connects everything to everything, and if it fails, travel can become complicated. Some member of your software group will need to roll Tap to find alternate, safe routes around the motherboard. If they fail, they will end up in a random location.
9	**Corrupted File Allocation Table:** This causes random software to disappear, and each software item loses a command from its hardware.
10	**Hard Drive Needle:** The hard drive needle has become uncalibrated and is generating bad sectors when writing. All of your items receive a bad sector.
11	**Power Supply:** If the power supply fails, everything else fails, but older power supplies are very resilient. What will happen is that it will overheat due to the loss of its fan. Once the power supply fan itself fails, this result no longer has an effect.
12	**Memory DIMM:** One of the memory DIMMs fails, forcing the rest to work harder and generate more heat. The temperature rises 1 point, and you roll again.

(¬) Scheduled Tasks

Scheduled tasks, or cron jobs, are tasks that the server performs periodically, such as emptying the recycle bin every night or sending a server status email every hour.

So, they can be terrible if they empty the recycle bin while you are logged in, or a chance to escape if you convince the email-sending script to attach the server status email.

To create your cron jobs, you must define three aspects:

- **Action performed:** This could be anything you can think of, from emptying the trash, which would cause the software inside to run away, to overheating the processor by creating a financial report from the company's accounting system that raises the temperature by 1 grade.
- **Location of the computer affected:** A task can affect one or more areas at the same time and at different times. For example, a cron job backs up the primary hard drive to a secondary drive and empties all its contents from the primary hard drive. Your software could face a deletion on the primary drive or an avalanche of data that crashes the secondary drive.
- **Trigger that activates it:** Although cron jobs are periodic, since we do not control real-time data within the server, we set triggers that activate the task, such as «when an even number of resets are made» or «when the temperature reaches 3».

Your software may need to trigger a cron job and have it perform the task it is supposed to. So you should be able to consult the list of scheduled tasks located in the /root directory and thus know how to launch the scheduled task.

```
I swear I was on a server with a scheduled
task that closed the CD-ROM cover every 5
minutes so it would not be used as a
coaster. -> updateDatos.sh
```

(°Δ°) Server Dwellers

There are a wide variety of software processing on the server, all of them trying to survive as best they can.

Some will try to escape, like your software, and others will try to stop you. you will have to interact with all of them by fair means or foul, convince them to let you through or intimidate them into doing so, exchange commands or steal them, compete for resources, or even kill their processes and loot their data scraps.

You can create these secondary software as your own software, or simply as an obstacle they must overcome through rolls.

(◣_◢) Horrors in FAT32

And if it were not already difficult to survive on a server about to crash, there are horrors lurking in the darkest sectors of the hard drive and among the corrupted sectors of memory.

Antivirus

Antivirus programs are the server's police and control the quarantine sector. Their main power is their numbers, and they will always come in groups of three or four to try to stop your software. They usually patrol storage drives and raid key elements like memory or BIOS, but you will never find them hiding in the recycle bin.

By permission, they are the only ones allowed to enter and exit the quarantine sectors (the prison inside a computer) and, therefore, to insert and remove software they consider dangerous.

They try to manage this software in a very fascist way. At the slightest opportunity, they will put you in quarantine without evidence or trial and can keep you there for cycles and cycles without telling you what you are accused of, asking you endless IF/ELSE questions that seem like infinite loops.

```
AAVAB | All AntiVirus Are Bastards ->
HelloWorld.bin
```

They tend to be quite hostile to free software and proprietary developments and treat proprietary software better, but if they discover you are pirated, they will likely arrest you to find out where you came from.

Their virus capture rate is ridiculous, and most of their arrests are usually false positives. In fact, most of the software in quarantine is free software and proprietary developments that have been unjustly stopped.

Infinite Loop

These vortices, whose flawed logic does not allow them to end, can suck you inside and trap you forever. You can represent them as energy tornadoes that attract nearby bytes and grow as they feed.

$ Each absorbed software increases the vortex's die level by one level, making it more powerful and attracting more software, and so on in a never-ending cycle. They start with a d4, and each time they swallow something important, they increase their die level. When they reach d12, they consume so many resources that they generate 1 temperature point.

It could be that your software want to feed it so it grows and generates more heat on the server, or they are trying to attract a virus so it can get closer and be devoured by the loop.

The only way to stop one of these vortices would be to turn the location where it is located on and off, for example in the BIOS, or cut off access to that location and leave it without any software to devour.

Firewall

More than a monster, we are talking about an obstacle, an extra security wall put in place by the system administrator. It can affect the entire system, like a DMZ that disconnects it from the rest of the network, or simply a barrier that protects specific sectors of the storage drive.

When defining a firewall, we must establish what it protects and how it can be traversed, since there is always a way to get through. Perhaps you need to be on a list of allowed scripts in the /root directory, or only open it in certain situations that your software must enforce if it wants to pass.

Malware

Malware is a program with secret intentions seeking to profit its creator, perhaps stealing passwords or emails or encrypting a storage drive to demand a ransom.

Malware is typically ordinary software: in fact, they often do not even know they are malware, and it activates in a certain situation. they are a kind of sleeper agent that, when given the command, performs some type of malicious action.

You can create malware as if it were any other software, but with an extra element called a Trojan. Trojans work a bit like scheduled tasks. When triggered, they perform a task that seeks to harm the system in a specific location. Here are some examples:

== If they manage to enter the /root directory and are left alone, they will begin encrypting its contents, rendering the entire system unusable. * If it finds a .cfg file and uses a «cat» command, it will attempt to break it down and search for keys and passwords within its bytes.

$ Unlike viruses, they can not be detected with a «diff»: it is the same software all the time. Still, their behavior can give them away, as they try to follow their trigger, even if they do not know why.

^(;,;)^ Virus

A computer virus is a malicious program that spreads onto a computer to damage it or steal information. It is very similar to malware, but it differs in how it works. While malware tries to hide and go undetected until it activates, a virus seeks to stealthily infect the entire system. Once it has sufficient power, it can be discovered and take control of the entire server.

Infected Software

When a piece of software comes into contact with a virus or an infected person and fails a Storage roll (to refuse to load the infection into its code) or Encryption roll (to protect itself from the intruder), it can be infected and will slowly become the virus itself.

A virus only has one chance to infect software. If it passes the resistance roll, the software becomes immune to the virus, and further contact with infected people from that virus is unaffected.

I do not put my code in places where other code has been. You never know what you might catch. -> virtual_desk.bas

Stages of a Viral Infection

During the infection process, it will be almost impossible to distinguish whether or not it is infected: only a Diff command can do so. In the early stages of the infection, it will continue to look normal, but nothing of the old software will remain inside, only the virus's programming, seeking to infect more software.

When your software is infected, it goes through several stages, and each stage is completed by performing a re-hash.

Step 0. It has just been infected. It is not contagious, nor is it detectable in any way. The infection is curable, and the infected person does not notice anything.

Step 1. After the first re-hash, the infection remains undetectable, but it can now be contagious. The infection is still curable, and the software still knows nothing.

Step 2. After the second re-hash, it remains infectious and is now detectable, but the virus has already taken control of the infection, and it is impossible to cure it: all that remains is to purge it. The software starts to notice strange things, but it still does not know for sure if it is infected.

Step 3. After the third rehash, it has been taken over by the virus and becomes a virus, controlled by the GM. It can continue to act normally and infect people, or it can turn into a virus and start destroying everything.

Battling a Virus

When the infected person completes their transformation, they become the virus itself. The virus can maintain its normal form and stealthily infect more scripts, or it can transform into a monster that actively seeks to corrupt more software or destroy the system. Your software is no match for these monsters, especially if the infection has spread.

The way to combat it is with cunning, good planning, and exceptional performance. Perhaps place a well-baited secondary storage drive and, when all the viruses enter, clog their incoming stream with large blocks of data so they can not escape.

Stackoverflow_

X

Stackoverflow_» is a two-session adventure for four software.

Your software run on an old computer in a nuclear silo from the 1980s/90s. The server is inside a DMZ, which prevents data and processes from entering or leaving. Geographically, we would say they are on a lost island in the ocean, and they do not have a boat.

But that is a lie: there is one way out, one way out. When the server's temperature reaches 9, a high-priority scheduled task is triggered, and the DMZ sends a warning email to the system administrator to intervene. If they attach themselves to the email, they can escape from the server, or perhaps even jump from the network card to another, more modern computer in the silo when the DMZ opens.

COMING SOON (April 2025)

Apendix

Software charsheet

Filename.ext
[₪]
[.*]

Storage	[d__] [d__]	Migration	[d__] [d__]
Encrypt	[d__] [d__]	Point	[d__] [d__]
Process	[d__] [d__]	Interface	[d__] [d__]

[>_]

[d__]
[d__]
[d__]

[↱]	[d12] [_]	[☰]	[d12] [d__]
[+]	[_]	[▩]	[_] [_] [_] [_]

Filename.ext
Origin

Skills

Storage	[d__] [d__]	Migration	[d__] [d__]
Encrypt	[d__] [d__]	Point	[d__] [d__]
Process	[d__] [d__]	Interface	[d__] [d__]

System commands

[d__]
[d__]
[d__]

Overclocking	[d12] [_]	Search for Processes	[d12] [d__]
Backup	[_]	Bad Sectors	[_] [_] [_] [_]

Server charsheet

Temperatura [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

Locations

BIOS	[d__] _____	Ports	_____
/root directory	[d__] _____	Quarantine Sectors	[d__] _____
Memory	[d__] _____	Network card/Modem	_____
Recycle Bin	[d__] _____	Storage Unit	[d__] _____
Motherboard	[d__] _____	Removable Storage Drives	[d__] _____

Scheduled Tasks

Failures

Terms You Can Use

This list of terms is a strange, yet very good-sounding concept that you can use when you need to explain what is happening or why something is failing.

== Load Balancing/Load Balancer
== Stack Overflow
== Iterations, Loops, and Conditions
== Patch and Update
== Memory Position
== Backdoor
== Cyclic Redundancy
== Replication
== Expired Token
== Virtualizer

Phrases like «Virtualizer is crashing» or «The load balancer is unbalanced» do not mean anything, but they seem like very computer science-related things.

License

THIS GAME IS BREATHLESS