BORN TO DIE . WORLD IS A FUARRRK

LAINCHAN.JP

鬼神 Kill Em All 1989 . I am trash man

Lainzine

410,757,864,530 oz. Soykaf

issue

レインジン

ALTHOUGH I MUST BE INNOCENT,
何の罪も無いはずなのに
I STILL FACE PUNISHMENT.
何らかの罰を受けてる
THOSE SEEDS WERE NOT SOWN BY ME, BUT NEVERTHELESS
時分で蒔いた種でもないのに
I SHALL UPROOT THE FLOWERS THEY WILL BLOOM.
咲き乱れた花摘まされる

no. 4

For all the Lains out there,
issue 4 of the Lainzine.

Published using Scribus.
All fonts are free,
all image work GIMP,
all vector work Inkscape.

Supplimental image credits:
Lainchan.org
NASA/ESA
Wikipedia

Go FOSS and multiply...
by the people, for the people.

Contents

COLOPHON

STAFF

LET'S JUST SAY YOU WERE TO CUT YOURSELF OPEN ONLY TO DISCOVER NOTHING BUT FLYING SPARKS, BROKEN CIRCUIT BOARDS, AND FRAYED WIRES...

...DOES THAT MEAN WE SHOULD JUST IGNORE YOUR DESPERATE PLEAS FOR MERCY?

ROBOTS CAN BE PEOPLE TOO. SUPPORT ROBOT RIGHTS.

this public service announcement is brought to you by cyberpunkforums.com

Editor's Note
By Junk / Stopwatch

Afternoon, morning, or another time.
Here's the fourth release of the lainzine, delivered exactly when I said it would come out: Soon.

My excuse for a late release this time is a mix of work, school, and not prioritizing lainchan as highly as I did when Kalyx was here. My work, I should add, was direct engagement with the 2016 United States election. The world is a very different place now and I hope that you are all safe in the upcoming future.

Stay tuned for a zine (unaffiliated to us) that some lains are putting together on opsec. Currently, they're looking for artists and formatters. If you can help, feel free to get in touch with anontrust@cock.li   `

6:10:1:14 6:12:4:26 57:3:4:2 56:4:6:1 61:11:2:23 60:3:2:13 60:1:5:12 58:1:3:2 31:8:1:22 29:1:1:7

--Junk

---end editor's note---

Stopwatch additional:

This issue is a full bumper packed 64 pages of Lain magic. Big thanks for all contributions to the Lainzine content and construction. This edition has been a long one in the making... As for the future of the zine, I intend to continue working on its layout and with editorial help will be looking for a release schedule somthing closer to tri-yearly, as opposed to the current try-yearly. The zine is not dead!

As many will know, Lainchan has recently (*cough*) sold out...

I would like to thank Kalyx for all the work and time he spent constructing this small corner of the internet and furnishing it so very well! Obviously the sale and transfer of Lainchan raised questions about its direction and future - but as it transpires, after the storm the sea is smooth. The current owner of Lainchan is now Appleman1234 and it appears he is not casual. Thanks for taking on Lainchan, and all the time you must now spend adminin' it. You seem to be willing and well able to keep it moving in a positive direction.

Now I present for your enjoyment in the following pages, Lainzine 4...

>Message ends.
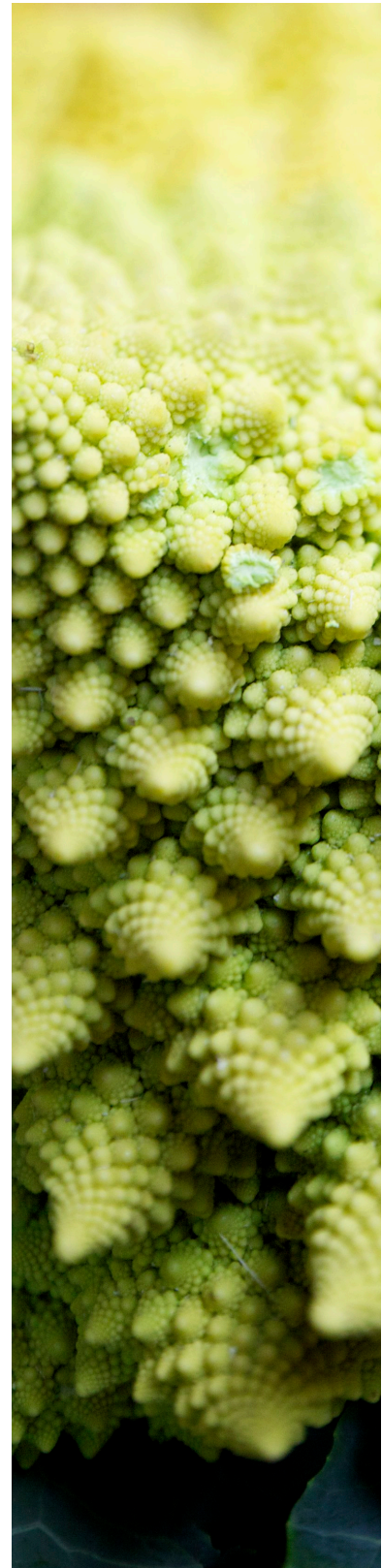
Chaos
By oceanhead

Summary

We study a simple system, called the Baker's Map, and we show that it exhibits all the characteristics of a *chaotic system*. Symbolic dynamics for the system will be constructed through the use of binary numbers. The system will be used as a model of a chaotic dynamical system for future articles.

No prerequisites are needed other than highschool maths so all the additional mathematical tools and ideas used will be constructed throughout the article.

Sections:

1) Some Prerequisites

What we want to do in this short series of articles is to show the existence of a certain, very intresting behaviour in simple, abstract physical systems. This behaviour is called *chaos*, and is not an exceptional phenomenon: it is quite generic and we experience it in everyday life.

The context in which chaos fits best is called the theory of *dynamical systems*. This theory is at the crossroad between maths and physics, the discipline called *mathematical physics*. A dynamical system is the abstraction of a physical system. A dynamical system is essentially made of two parts, a set, called *phase space* or simply space, and a function, called the *dynamics*, that associates a point in the space to another point in the space. What we mean for space is allready a complicated matter, and we won't dive into it. For what we need, it suffices to think of the space X as a subset of space, $X \subset \mathbb{R}^n$. The dynamics is just a function $\Phi : X \to X$. This function is called dynamics because we want to think about it as a model of the motion of a particle: to do so, we think of a point x in X as a particle, and of $\Phi(x)$ as the point in X to which x has moved to after a fixed time. The whole motion is calculated applying iteratively the dynamic map Φ. For this reason, a rule that associates an "initial point" x to its transform after k units of time remains defined:

$$x_{k+1} = \Phi(x_k)$$

This means: chosen a time step t_0 , the particle that was in x_k at time $t = kt_0$, will be at time of $t = (k + 1)t_0$ in position $x_{k+1} = \Phi(x_k)$. This also means that if we want to know the position x_k of a particle that starts its motion in x_0 after a time of $t = kt_0$, we need to apply Φ to x_0 k times:
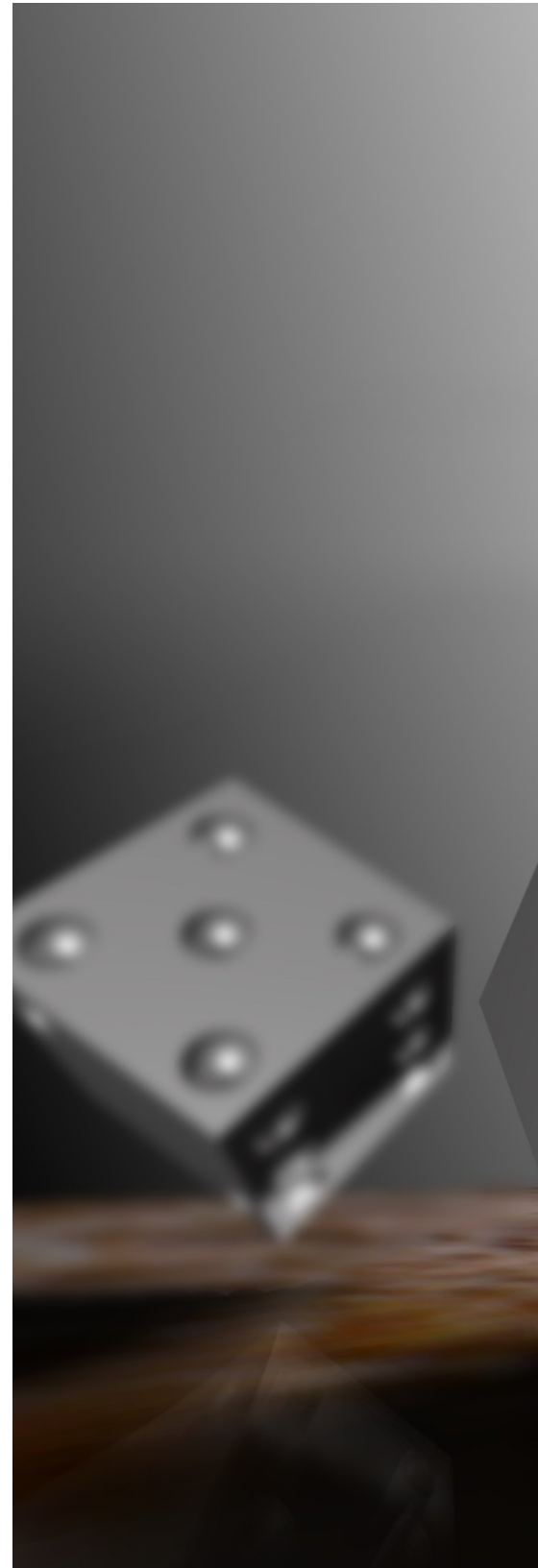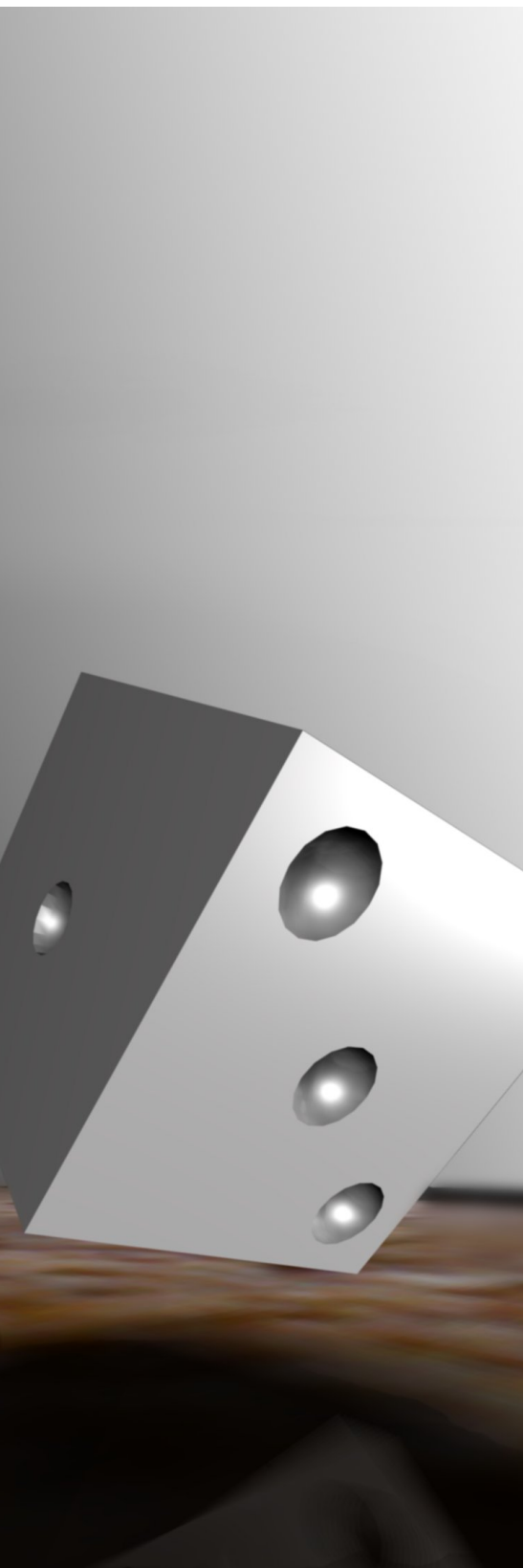
$$x_k = \Phi^k(x_0) \text{ where } \Phi^k = \underbrace{\Phi \circ \dots \circ \Phi}_{k \text{ times}}$$

So this is the picture: we take a point $x \in X$ and we consider the subset

$$\mathcal{O}(x) = \{\Phi^k(x),\, k \in \mathbb{Z}\} \subset X$$

called orbit of x, and we think of the integer $k \in \mathbb{Z}$ as a time unit. What we want to do when analyzing a dynamical system is to describe qualitatively what the dynamical map does to the points in phase space X.

2) The Baker's Map

2.1) Presentation of the system

We now concentrate on a single, simple dynamical system, called the Baker's Map.

The phase space of the system is the square of unit lenght (with the edges removed), $X = [0, 1[^2$, which we will consider with cartesian coordinates:

$$X = \{(x, y) \in \mathbb{R}^2 : 0 \le x < 1,\, 0 \le y < 1\}$$

The dynamical map might seem a little complicated but with some figures it can be explained very easily. This is its form:

$$\Phi(x, y) = \begin{cases} (2x, \tfrac{1}{2} y) & \text{if } x < \tfrac{1}{2} \\[2mm] (2x - 1, \tfrac{1}{2} y + \tfrac{1}{2}) & \text{if } x \ge \tfrac{1}{2} \end{cases}$$

To visualize it, consider Figure 1. It is a plot of 50 000 random points in X, centered in $(\tfrac{1}{2}, \tfrac{1}{2})$. Think of X as a square of something malleable, like dough. The Baker's map does what a baker does when working dough: you take the dough, you squash it down, you cut it in two and you put a piece on top of the other. If you are not convinced that this is what the map does, the figures will show you that this is exactly its action.

Think of the black ball as a dot of food coloring dabbed in the middle of the dough.
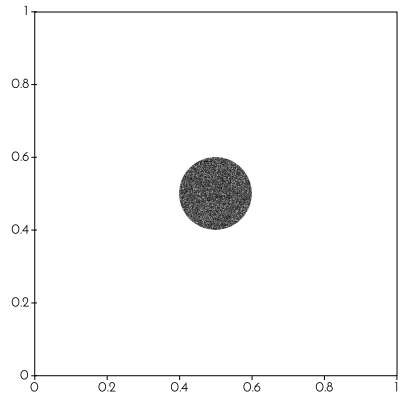
FIGURE 1: A ball of radius 0.1 of 50 000 random points in X

Note that the dot is exactly centerd in the square. This has consequences on the speed with which the ball will be transformed, because the map is piecewise: that is, it does different things left of $x = \frac{1}{2}$ then it does right. In Figure 2 you can find a schematic drawing of the action of the map on the phase space. The iterations of the Baker's map on the initial ball can be found in Figure 3. Think of the dough being squashed down untill it's twice as long and half as tall. Then it is cut in two, and the two parts are stacked. If you keep this in mind, the first iteration of the map will be clear to you: the ball is cut in two and half as tall. Now let's take the resulting phase space and let's apply the dynamics again, obtaining the second iteration. This time no part of the black dot has $x = \frac{1}{2}$, so the cutting has no effect on it. The result is that the black dot is *only squashed*, and it progressively gets longer and closer to the middle. This goes on for a while, untill the two "limbs" are squashed and elongated enough that a piece of them intersect the half. We can jump to the fourth iteration, where a qualitative change ensues. This is when the magic starts! The limbs get repeatedly cut and stacked together, forming long and progressively finer filaments that, step by step, cover the whole phase space, making the points sparcer and sparcer. After 10 steps there is still a glimpse of regularity: we can still distinguish the filaments. This form goes on for some time, depending on the number of points that make the initial ball. The next steps make the points limbs lose their structure, distributing the particles throughout the dough. In this case, the phase space will go under another "transition" after 10 steps, in which the filaments are completely disintegrated.

The ball structure is completely lost. The points from now on wander around the dough, and no qualitative change ensues from now on. This is the situation of iteration 20.

It is clear that something "chaotic" is going on: after some steps, any "definite lump" in phase space will loose all its shape and will have its points distributed throughout the phase space (this is why bakers use[1] the baker's map to kneed dough: all the lumps get distributed in the dough and after some steps the dough is allmost perfectly uniform). What we want to show in the next section is that even if the system tends to rip up and uniform shapes in phase space, it is *deterministic*.

2.2 Symbolic Dynamics for the Baker's Map

Let's focus on the action of the Baker's map on one point. Can we predict where it will go after k iterations? The first idea is to take its coordinates, apply the map, then plug the result in the map, and repeat k times. This surely will show the point's coordinates after k iterations. But this is not an intresting way to do it, because we would like to *predict* without having to do all the calculations.

There is actually a way to predict the behaviour of one point's orbit, more pre-cisely, there is a way to *find* a point that will spawn an orbit that, iteration after it-eration, will pass *infinitesimally close* to points we have *chosen*. To see this, we have to find smarter coor-dinates on phase space that are adapted to the map's action, equivalently, coordinates in which the map's action is extremely easy.

To do this, we must recognise what the map essentially does. As we have said, apart from the cutting, the map *stretches* in one direction and *squashes* in the other, and it does this by a factor of 2. Stretch-ing by a factor of 2 is done by multiply-ing by 2, while squashing is done by divid-ing by 2. So the essential part of the map a division by two in one

(a) First iteration

(b) Second iteration

(c) Fourth iteration

(d) Sixth iteration

(e) Tenth iteration

(f) Twentieth iteration
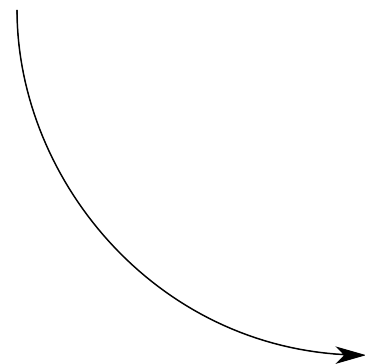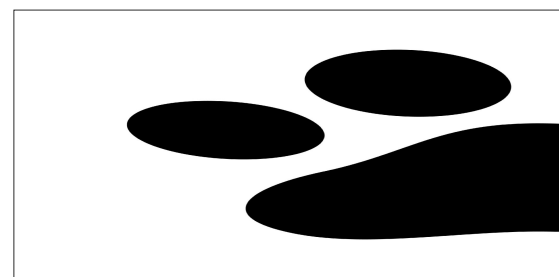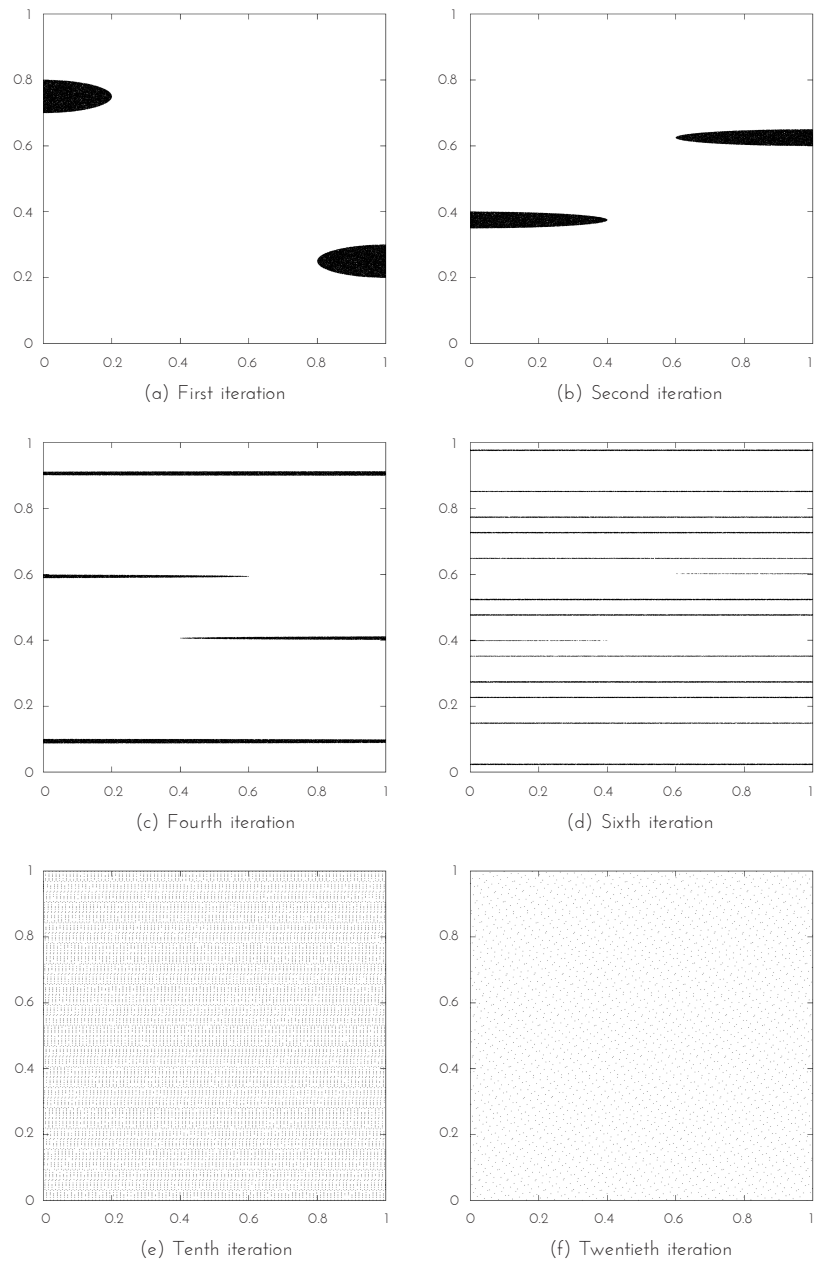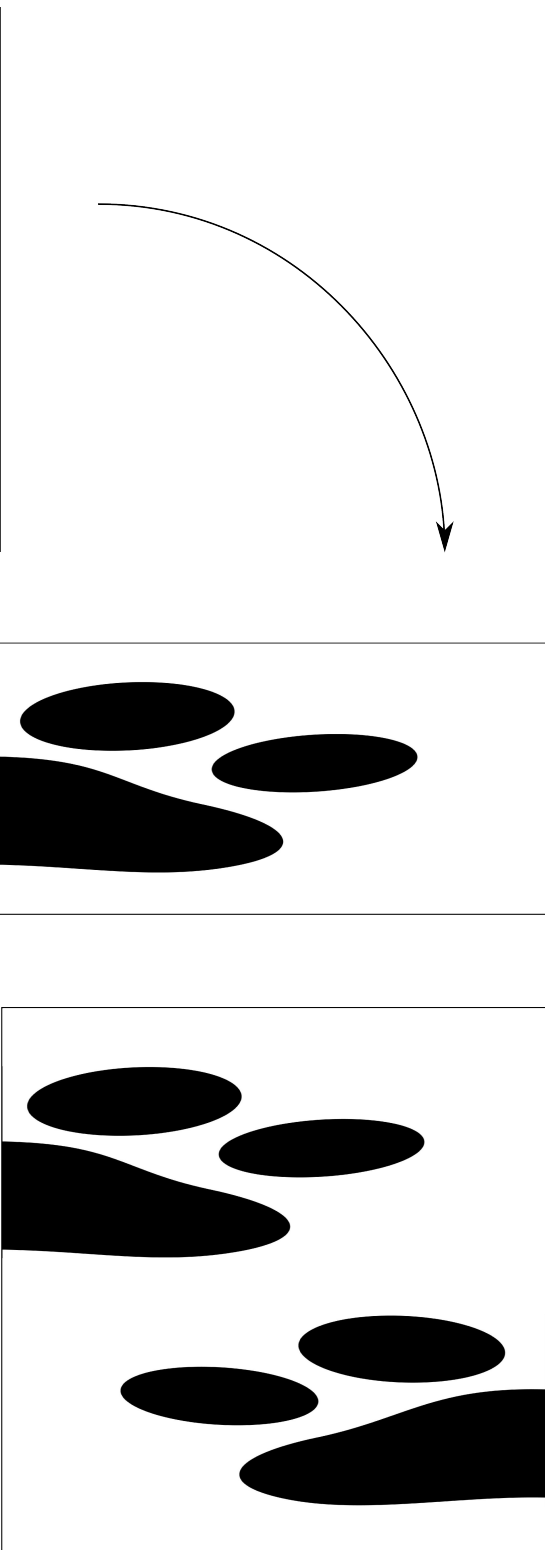
FIGURE 3: Iterations of the Dynamical Map.

direction, and a multiplication in the other. This gives us an idea: in base 2, multiplying by 2 is equivalent to adjoining a 0 at the end of the number, equivalently, shifting towards the right the decimal point, while dividing by 2 is equivalent to shifting towards the left the decimal point. Thus if we write the coordinates in X in binary notation we can capture the stretching/squashing propriety of the map easily. This is actually an intermediate step, because we need to take account of the "cutting in two" part of the map. A deeper analysis of the map shows that the best co-ordinates[2] possible are these: you take the binary expansion of the point (x, y)

$$x = 0.a_0 a_1 a_2 \ldots \qquad y = 0.b_0 b_1 b_2 \ldots$$
$$\text{where } a_k, b_k \in \{0, 1\} \ \forall \ k \in \mathbb{Z}$$

Note that these expansions are not necessarily finite, and if they are we will think of them as infinite by adjoining an infinite sequence of zeroes, in this way:

$$0.\alpha_0...\alpha_n = 0.\alpha_0...\alpha_n 0000...$$

Then[3] you take the binary expansion of y, reverse it and "attach" it to the binary expansion of x, this way:

$$x = 0.a_0 a_1 a_2... \qquad y = 0.b_0 b_1 b_2... \implies$$
$$\sigma = ...\sigma_{-2}\sigma_{-1}.\sigma_0\sigma_1\sigma_2\cdots = ...b_2 b_1 b_0.a_0 a_1 a_2...$$

So now, instead of phase space X and it's points we will consier the set

$$\Sigma = \{\sigma = ...\sigma_{-1}.\sigma_0\sigma_1 \cdots : \sigma_k \in \{0, 1\} \ \forall k \in \mathbb{Z}\}$$

of double-infinite "strings", or "words", with "symbols", or "letters", in $\{0, 1\}$. The corrispondance is one on one, because all we done is to find a compact way to write a point in X, and it is actually a function

$$h : X \longrightarrow \Sigma$$
$$h(x, y) = ...\sigma_{-1}.\sigma_0...$$
$$\text{where } x = 0.\sigma_0\sigma_1... \ , \ y = 0.\sigma_{-1}\sigma_{-2}...$$

For example, lets consider the string associated to the point $(\frac{1}{2}, 0)$. The binary expansion of $\frac{1}{2}$ is 0.1, so $h(\frac{1}{2}, 0) = ...00.10...$

Now we need to find a map $\Psi : \Sigma \to \Sigma$ that does what the Baker's map does on X.

To do this we must understand what the binary expansion tells us about the points in phase space. Lets focus on the x component. Any x such that $x < \frac{1}{2}$ will have binary expansion starting with 0, that is $x = 0.0...$, while any x such that $x \geq \frac{1}{2}$ will have binary expansion starting with 1, $x = 0.1...$. This means that any point right of the center $(\frac{1}{2}, \frac{1}{2})$ will be of the form $...\sigma_{-1}.0\sigma_1...$ and any point left of the center will be of the form $...\sigma_{-1}.1\sigma_1...$. The same thing can be said about the y component, switching right with under and left with over. This means that if we divide in four quadrants our X, the letters in position -1 and 0 tell us in which quadrant the point is. This reasoning can be repeated with the next two letters of the string, dividing each quadrant in additional four quadrants, and so on: the letters in position k and $-(k+1)$ tell us in which square of a grid of 2^{2k} squares in X the point is. See figure 4. This is essentially the meaning of a binary expansion, so it is nothing new but a different point of view. So a string localizes a point in X with precision growing with the number of letters of each string we consider. This will be important for prooving some fun properties peculiar to the Baker's map.

Recall that the Baker's map is defined by

$$\Phi(x, y) = \begin{cases} (2x, \frac{1}{2}y) & \text{if } x < \frac{1}{2} \\ (2x, -1, \frac{1}{2}y + \frac{1}{2}) & \text{if } x \geq \frac{1}{2} \end{cases}$$

We now proove that

if $\sigma = h(x, y)$ then $\Psi(\sigma) = \sigma'$
where $\sigma'_k = \sigma'_{k+1}$

FIGURE 4 (pt1): Phase space partitions and letters in strings.

X ...1.0...	...1.1...
...0.0...	...0.1...

X			
...11.00...	...11.01...	...11.10...	...11.11...
...01.00...	...01.01...	...01.10...	...01.11...
...10.00...	...10.01...	...10.10...	...10.11...
...00.00...	...00.01...	...00.10...	...00.11...

FIGURE 4 (pt2): Phase space partitions and letters in strings.

that is, on the word set Σ the Baker's map acts as a translation towards the right of the decimal point:

$$\Psi(\ldots\sigma_{-1}.\sigma_0\sigma_1\ldots) = \ldots\sigma_{-1}\sigma_0.\sigma_1\sigma_2\ldots$$

This is actually very easy to proove, infact, we allready prooved it. To understand this lets concentrate on the points with $x < \frac{1}{2}$. Their associated strings in word space are the strings of the form $\sigma = \ldots\sigma_{-1}0\sigma_1\ldots$. Since $x < \frac{1}{2}$, the first part of the map applies, so x is multiplied by 2 and y is divided by 2. Notice that since $y < 1$ after the division $y' < \frac{1}{2}$. On string space this means that $\sigma' = \ldots0.\sigma'_0\ldots$. The value of σ'_0 depends on if $x < \frac{1}{4}$ or not: since σ'_0 tells us in which subquadrant of the first quadrant of X the point is, and since the x-component of such point comes from a multiplication by two, if σ_1 was 0 now σ'_0 is 0, and the same with 1. To finish, note that if $x \geq \frac{1}{2}$, then Ψ brings the 1 from x to y. Thus $\sigma' = \ldots\sigma_0.\sigma_1\sigma_2\ldots$. A straight forward calculation may be needed to grasp this fact fully. You may try with the points $(\frac{1}{4},\frac{3}{4})$ and $(\frac{3}{4},\frac{7}{8})$.

This result may be summarized by the concise formula[4]

$$\Psi \circ h = h \circ \Phi$$

These "coordinates" are called *symbolic dynamics* for the Baker's map, because instead of looking at the evolution of the system in phase space, from now on we will look at the evolution of strings in word space. This may seem useless at first, but in this way we will proove important properties of the system.

The first thing we will do is to use the symbolic dynamics to find points that spawn predetermined orbits. Consider this (weaker) problem: we want to find a point $P = (x, y)$ in phase space that after k iterations of Φ has $x > \frac{1}{2}$ (or $x \leq \frac{1}{2}$). This is a very easy problem: since Ψ acts as traslation of the decimal point on string space, every string σ that has $\sigma_k = 1$ (or 0) will have its k-th iteration with the letter 1 right before the decimal point, so it will certainly have $x > \frac{1}{2}$. Now, if we want to have $y > \frac{1}{2}$ too, then, for the same reasons, all strings with $\sigma_{k-1} = \sigma_k = 1$ will have their k-th iteration as wanted, because $\Psi^k(\sigma) = \ldots1:1\ldots$. The points we are looking for are then simply $P = h^{-1}(\sigma)$. This reasoning can be further extended to any finer partition of X in squares, in fact, if you find a string σ with a block of $2s$ predetermined letters in it starting from position $k - s$, then after k iterations of Ψ the string will have this block in center position, divided by the decimal point, thus the point $P = h^{-1}(\sigma)$ will be in the cell that is described by that block of letters.

The last generalization that can be made with the same reasoning is that instead of considering only a fixed number of iterations k, we could be intrested in points whose orbits "hop" in given squares of some partition of X. This is now easy to construct: if the partition of X has squares with sides of length $\frac{1}{2^s}$, all we have to do is to take a string that is in blocks of $2s$ letters that describe the wanted square, and let the Ψ map progress in steps of s. Every anti-image $P = h^{-1}(\sigma)$ at these fixed steps of s iterations will clearly be where we wanted it to be, realizing our intent. So if we had fixed some points in phase space, then this point will have its orbit pass as close as we want to these points, just choosing a fine partition of phase space and letting the orbit go in squares that contain the fixed points. This tells us a second meaning of the strings: they describe the *history* of a given point.

There are two very important things to be noted: in the above construction, if the point $P = h^{-1}(\sigma)$ were to be taken with a small error, then after a few iterations the orbit would diverge from the one predeter-mined by σ *exponentially fast*[5]. Somewhat similarly, if instead of a point we take a few points condensed in a small square of our fine

partition, then they will be scattered throughout the whole phase space, in an *uniform* manner. These two considerations will motivate the definitions of a dynamical system and a chaotic dynamical system.

2.3 Metric compatibility and the dense orbit

What we want to analyze is the so-called *metric structure* of X and Σ. A metric structure is, very informally, a way to mesure distances in a space. A little more formally, a metric on a set Y is a function $d : Y \times Y \to [0,+\infty[$ satisfying some more or less obvious requisites:

$$d(Q, P) = 0 \Longleftrightarrow Q = P$$
$$d(\mathrm{P}, Q) = d(Q, P)$$
$$d(Q, R) \leq d(Q, P) + d(P, R)$$

They mean: two points are distant 0 if and only if they are the same point, distance does not depend on which point you compute it, and the third is called "triangle inequality" and essentially it means that the length of one side in a triangle is less than the sum of the length of the other two. The last requisite is there to restrict the possible metrics to the ones that are at least similar to the usual Euclidean metric defined by Pythagoras' theorem (see below). The couple (Y, d) is called *metric space*.

In X there is an obvious metric: if P and Q are points, then we can define the distance between P and Q this way

$$P = (x_P, y_P) \qquad Q = (x_Q, y_Q)$$
$$d_X(P, Q) = \sqrt{(x_Q - x_P)^2 + (y_Q - y_P)^2}$$

which is just the Euclidean length of the segment joining P and Q. There also is a metric on Σ:

$$\text{define } \delta(\sigma_i, \sigma'_j) = \begin{cases} 0 \text{ if } \sigma_i = \sigma'_j \\ 1 \text{ if } \sigma_i \neq \sigma'_j \end{cases}$$

$$\text{then } \delta_\Sigma(\sigma, \sigma') = \sum_{k \in \mathbb{Z}} 2^{-|k|} \delta(\sigma_k, \sigma'_k)$$

Since the sum is infinite, the $2^{-|k|}$ term is there to make it converge, that is, to have it return a finite number for every two strings in word space. Ignoring the subleties, it is easy to see that such metric is well defined and is a real metric. Its meaning is that two strings are closest when they have the same symbols in the same positions (and so they are the same string). The $2^{-|k|}$ term also guarantees that the strings that differ only on terms with large k are still close.

Thinking about the meaning of the strings, you can see that this is a good thing.

Notes

[1] Actually, as far as I know, they use a different map, that in jargon is called Horseshoe map, in which the cutting is replaced by folding in half. Their behaviour is quite similar though, and they are both chaotic.

[2] To be precise, what we are considering is not a proper change of coordinates, since we are not changing the way we describe the phase space, but we should think of it as a *dynamical conjugation*, that is, we are looking for an equivalent dynamical system, in a certain sense. We will expand this point of view further in the next article, when we introduce the formal definition of a dynamical system.

So $(\mathrm{X}; d_\mathrm{X})$ and (Σ, d_Σ) are two metric paces: does the symbolic dynamic

$$h : \mathrm{X} \to \Sigma$$

send *close points* in *close strings*? In mathematical terms, we are asking:

if $\exists\, \varepsilon > 0$ such that $d_\mathrm{X}(P, Q) < \varepsilon$
then $\exists\, \delta > 0$ such that $\delta_\Sigma(h(P), h(Q)) < \delta$

The answer is no! As a matter of fact, the key is that we have eliminated from Σ all the strings that end (or begin) with an infinite succession of 1s, and that lets us get as close as we want to a point that has no image in Σ. But all is not lost, because it is true if we substitute h with h^{-1}, and that is the "direction" we are intrested in, since we want to proove things in string space and then transpose them in phase space.

The last thing we prove about the Baker's map is that there exists a point that spawns an orbit with a peculiar metric property:

$\exists\, P \in X$ such that $\forall\, R \in X\, \exists\, Q \in \mathcal{O}(P)$
and $\exists\, \varepsilon > 0$ that *satisfies* $d_\mathrm{X}(Q, R) < \varepsilon$

That is, the orbit of P passes *as close as we want to every point of* X. This is called a *dense orbit*. Another way of seeing this property is that if we draw point after point of the orbit of P in X, we eventually[6] fill the whole phase space without leaving any gaps.

As we have seen, the fruitful idea is to try to construct a string whose image under h^{-1} has the metric property we are looking for. To do this, note that given a generic string, if we truncate it left and right and we append zeroes to it, we obtain another string that is close to the beginning one, and we can decide how close such string is by deciding how many letters we truncate from the initial string. Then we note that we can order the set of such "finite" strings, for example in this way:

0 1 00 01 10 11 000 001 010 011 ...

Then we consider the string that is composed by an arbitrary semi-string on the left and on the right all the set of finite strings written one after the other, and the respective point in X:

$$\hat\sigma = \ldots\sigma_{-2}\sigma_{-1}.0100011011000001010011 \ldots$$
$$\hat{P} = h^{-1}(\hat\sigma)$$

We assert that this point's orbit is our dense orbit. Indeed, chose $\varepsilon > 0$ as small as you want, and let $k \in \mathbb{Z}$ be such that $2^{-k} < \varepsilon$. For every $Q \in \mathrm{X}$, let $\tau = h(Q)$. The block $\tau_{-k} \ldots \tau_{k-1}$ is certainly somewhere in σ, since it contains all finite blocks of letters. Suppuse it is in between positions $t-k$ and $t+k$-1 for some $t \in \mathbb{Z}$. Then after exactly t iterations of the dynamical map the block will be centered around the decimal point in the iterate of σ, and at that moment the iterate will be less than ε away from Q.

[3] We should eliminate all the numbers with an infinite sequence of ones at the end, such that the change of coordinates is *bijective*. This means that if the expansion were decimal, we would not distinguish between 1 and $0.\bar{9}$, which is a good thing.

[4] This is what we called dynamical conjugation earlier.

[5] The exponential character comes from the fact that every iteration multiplies x by 2 and divides y by 2.

[6] after an infinite amount of time

WORLD L.S.D

L✧MECH事業部サポート

L.S.D

一天四海
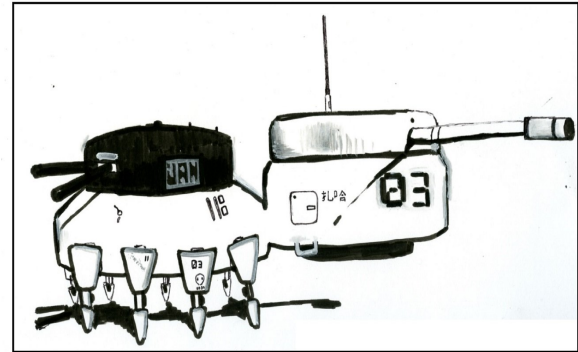
WORLD L.S.D (in illustrations):

Tom Millicent

LOMECH SUPPORT DIVISION'S PRIMARY OBJECTIVE WAS A **RUN AND GUN** MISSION ~ PROTECT THE **CITY** AT ALL COSTS! THE MILITARY SERVED ONLY TO **DROP** LOMECH UNITS WHERE THEY WERE NEEDED, DUE TO CORRUPTION ON A LARGE SCALE; THEY HAD BEEN INFILTRATED BY **JAH**, WHO WOULD STOP AT NOTHING. G.O.D WAS OVER POWERED ~ U-5 HAD ACCESS TO ITS MAIN-FUNCTION, THOUGH THIS DIDN'T ENSURE A MALEVOLENT FORCE COULDN'T **HACK** IT FROM AN EXTERNAL POSITION ~ AND THEY WERE DOING SO FREQUENTLY! **X89** EXPERIENCED ONE SUCH OCCURENCE OF **JAH_SECTOR'S** INTERFERENCE.

LOMECH SUPPORT DIVISION* JAH IS RE-ROUTING G.O.D'S ENERGY, ITS INTERNAL DEFENSES ARE AT CRITICAL LEVEL! ONE OF OUR UNITS IS COMPROMISED AND UNDER THEIR CONTROL - WE NEED LOMECH ON THE GROUND!*
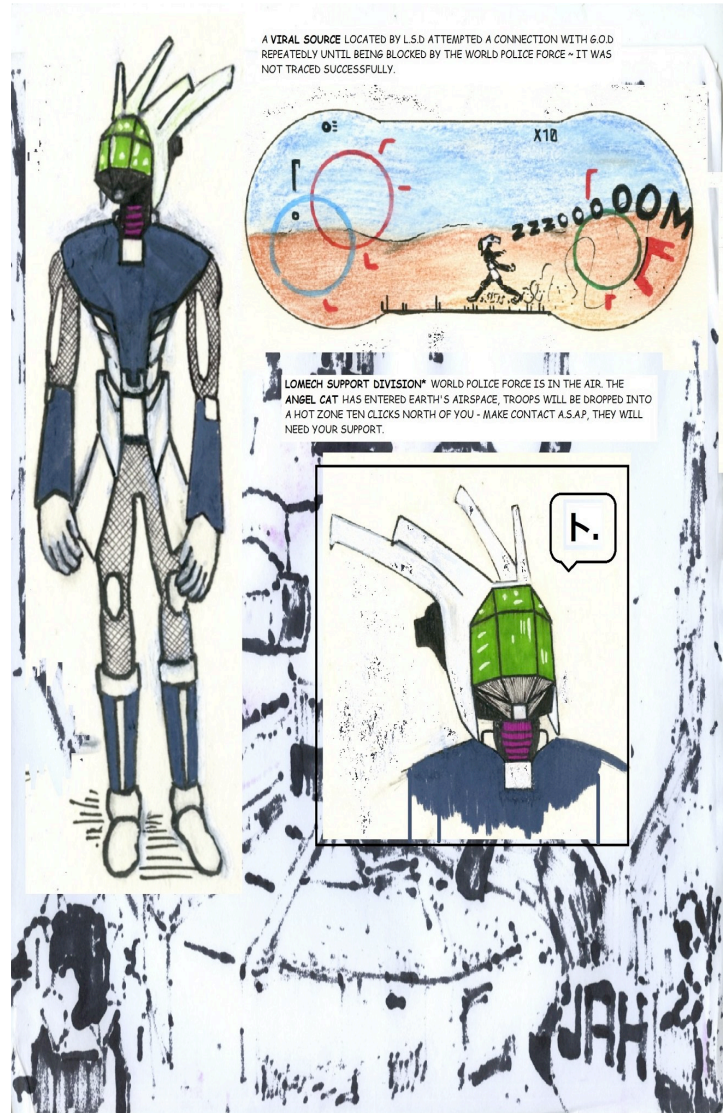
A **VIRAL SOURCE** LOCATED BY L.S.D ATTEMPTED A CONNECTION WITH G.O.D REPEATEDLY UNTIL BEING BLOCKED BY THE WORLD POLICE FORCE ~ IT WAS NOT TRACED SUCCESSFULLY.
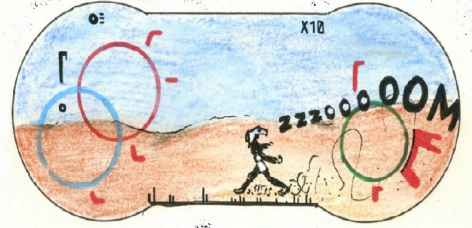
LOMECH SUPPORT DIVISION* WORLD POLICE FORCE IS IN THE AIR. THE **ANGEL CAT** HAS ENTERED EARTH'S AIRSPACE, TROOPS WILL BE DROPPED INTO A HOT ZONE TEN CLICKS NORTH OF YOU - MAKE CONTACT A.S.A.P, THEY WILL NEED YOUR SUPPORT.
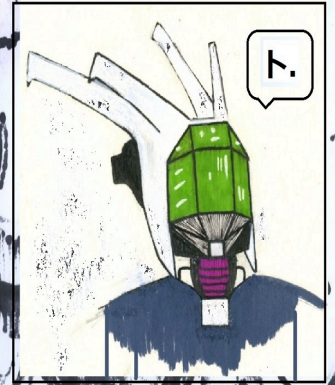
Compiler Optimizations

By Niles

1 Compiler Optimizations

The general idea of optimization is to replace code with more efficent code that has the same end result.

The problem with this is that the ANSI C standard specifies that the results of the new code have to match the original code in an "abstract machine." In an abstract machine, undefined behavior can never happened.

Undefined behavior in the C standard consists of a "shall" or "shall not" from the standard being violated, anything listed in the standard as "undefined behavior" or anything not listed in the standard at all.

The general problem is that sometimes programmers rely on checking for undefined behavior to make sure it has not happened. The compiler assumes that this undefined behavior can never happend and simply "optimizes" away the programmer's code.

2 Algebraic Simplification

Some example C code might look like this

```
char *ptr; // ptr to start of array
char *max; // ptr to end of array
size_t len;
// Other code
if (ptr + len > max)
        return EINVAL;
```

This makes sense assuming normal values of len, however if len is generated incorrectly or supplied by the user, the value of ptr + len may overflow, causing ptr + len to be smaller than max and allowing whoever controlls len to access arbitrary addresses in memory beyond the end of the array.

The common resolution to this is to write something like this:

```
if (ptr + len < ptr || ptr + len > max)
        return EINVAL;
```

Unfortunately, in this instance the compiler can apply algebraic simplification. When comparing P + V1 and P + V2 where P is the same pointer and V1 and V2 are the same integer type, the compiler optimizes it down to a comparison between V1 and V2. What this means for our program is that our check, ptr + len <ptr could be rewritten as ptr + len <ptr + 0, and that can be algebraicly simplified to len <0, which is impossible because len is unsigned.

This means that to our compiler, ptr + len <ptr is dead code because it will always return false, and gcc 4.3 with -O2 will "optimize" it away.

The only way to mitigate this is to change the check to something like this

```
if (len > max - ptr)
        return EINVAL;
```

3 Algebraic Simplification, cont.

Another example of algebraic simplification changing the result of code might look like the following

```
int i = (x * 1000) / 2000
```

A compiler would usually optimize this to i = x / 2, however if x * 1000 were going to overflow, this does not do the same thing. For example, if x were set to 1073742, then the non-optimized program would set i to -2147483, however the optimized code would return the mathematically correct value of 2147484.

4 Integer Overflow

Consider the code

```
int f() {
        int i;
        int j = 0;
        for (i = 1; i > 0; i += i)
                ++j;
                return j;
        }
}
```

Eventually the loop will end because i will overflow, however as the compiler says integer overflow is undefined behavior and can therefore never happen, gcc 4.3.2 with -O2 "optimizes" this code to an infinite loop.

This type of check even exists in the GNU C Library, where the implimentation of mktime will always return as if a time adjustment was sucessful, even though the program has checks built in to return -1 if it were to fail due to integer overflow.

5 Loop Hoisting

The compiler is allowed to pull partial (or even sometiles full) statements from inside a loop to outside the loop. This is useful if you are, for example, adding a complex numerical expression to a final result, but the variables in the expression are not modified in the loop. In that case, something like
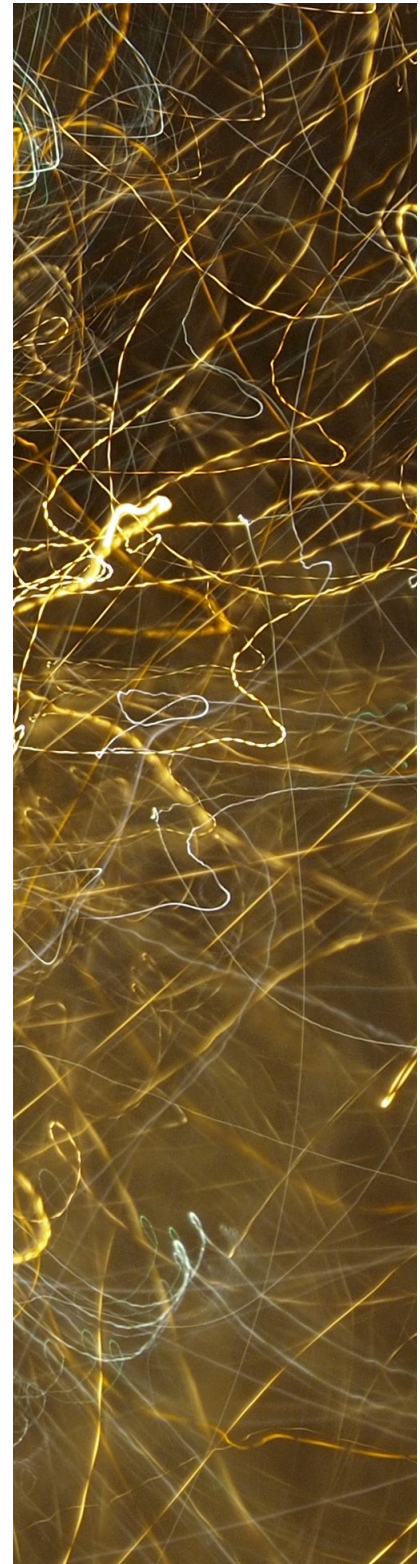
```
for (int i = 0; i < 10; i++) {
        total += a*x^15 % i;
}
```

may be optimized into the much faster code

```
int temp = a*x^15;
for (int i = 0; i < 10; i++) {
        total += temp % i;
}
```

The unfortunate side effect of this is that it's not just pulling constant expressions outside of loops, it can also rearrange the order of statements inside a loop for optimal effcency. For example, in the following code

```
signed int si1 = atoi(argv[1]);
signed int si2 = atoi(argv[2]);
signed int result = 8;
size_t i;
puts("log message one.\n");
for (i = 0; i < MAX; ++i) {
        puts("log message two.\n");
        if (argc == 8) i++;
        result += i + si1 % si2;
        puts("log message three.\n");
}
printf("Result = %d.\n", result);
```

In this case, the program should print "log message one.," then "log message two.," then fail due to a floating point exception. The optimized program, however, only prints the first message before failing, leading to some serious head-scratching debugging hell.

6 Clearning Sensitive Memory

By far the largest optimization a compiler can make is dead code removal. For example, a malloc call that creates an area of memory that is then never used can be removed from the program before it is compiled without any resulting changes. However, this is not always the best option for security reasons. For example, if the user types a password into the program, it needs to be stored in memory. After the password is done being used, it should be overwritten with either null bytes or random data so that when the memory is reallocated to another process later, the second process cannot see the user's password. An example is shown in the following code

```
void getPassword(void) {
        char pwd[64];
        if (GetPassword(pwd, sizeof(pwd))) {
                /* check password */
        }
        memset(pwd, 0, sizeof(pwd));
}
```
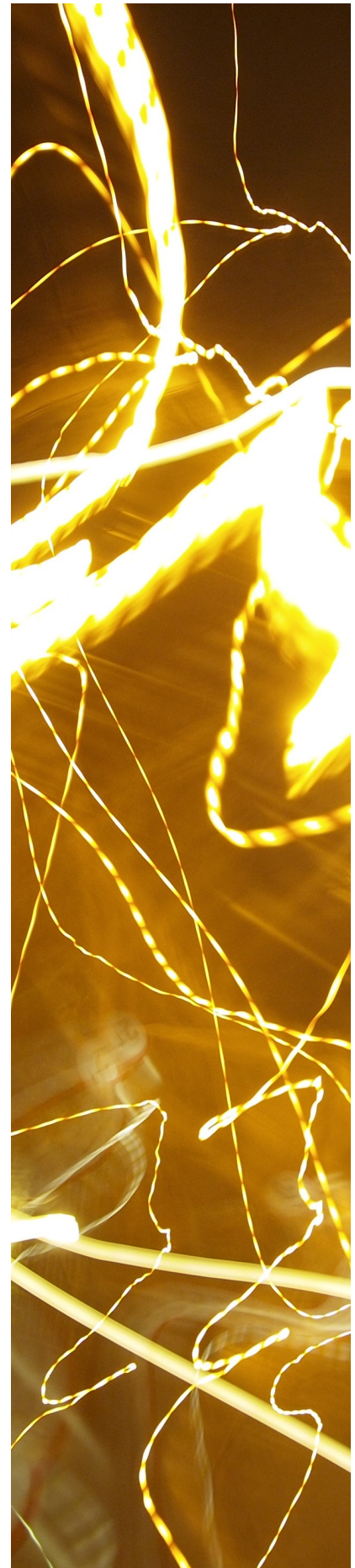
However, if this memory is never used after the call to memset, the compiler can just regard the memset instruction as dead code and elade it. This is an obvious security flaw, but there are several ways to mitigate it. The first to be implimented was the Microsoft Visual C function ZeroMemory().

However, it sometimes gets optimized out as well, so they added a new function, SecureZeroMemory(), which the compiler is not allowed to optimize out.

Another solution is to surround the code with #pragma directives, like this

```
void getPassword(void) {
        char pwd[64];
        if (GetPassword(pwd, sizeof(pwd))) {
                /* check password */
        }
#pragma optimize("", off)
        memset(pwd, 0, sizeof(pwd));
#pragma optimize("", on)
}
```

The pragma directive is supported on some versions of Microsoft Visual Studio and may be supported on other compilers.

Another common home-baked solution is an abuse of the volatile keyword, like this.

```
memset(pwd, 0, sizeof(pwd));
*(volatile char*)pwd = *(volatile char*)pwd;
```

However, depending on the implimentation this sometimes zeroes the whole buffer but sometimes zeroes only the first byte, leaving the remainder intact.

The final solution is to write a secure memset function that uses the volatile char trick across the whole buffer, something like this

```
void *secure_memset(void *v, int c, size_t n) {
        volatile unsigned char *p = v;
        while (n--)
                *p++ = c;
        return v;
}
```

The problem with this is that not all common compilers always respect the volatile qualifier. On top of that, this prevents the code from being optimized at all, as most compilers usually replace memset with a few assembly instructions that are much more efficent.
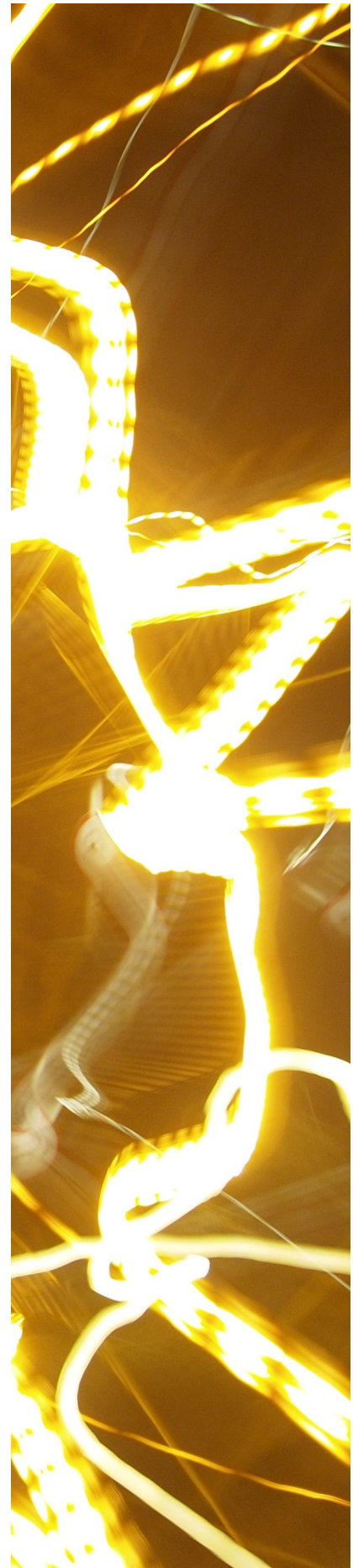
7 The Volatile Qualifier

Take the example code

```
volatile int buffer_ready;
char buffer[BUF_SIZE];
void buffer_init() {
for (size_t i = 0; i < BUF_SIZE; i++)
        buffer[i] = 0;
        buffer_ready = 1;
}
```

In this case, it would seem that buffer_ready should be immune from common optimization problems. However, because the for loop does not access any volatile locations or modify any related variables, the compiler can move the buffer ready = 1; line above the loop, defeating the developer's intent.

8 Optimizing for Embedded Systems

The following C code is a function that resets a watchdog timer in a hypothetical embedded system

```
extern volatile int WATCHDOG;
void reset_watchdog() {
        WATCHDOG = WATCHDOG; /* load, then store */
}
```

Regardless of the optimization level, a conforming compiler must convert this to code that loads and then stores the watchdog register.

Recent versions of GCC for IA-32 emit this assembly code

```
reset_watchdog:
        movl WATCHDOG, %eax
        movl %eax, WATCHDOG
        ret
```

However, the latest version of GCCs port to the MSP430 microcontroller compiles the code into the following assembly:

```
reset_watchdog:
        ret
```

Thus the compiler's optimizations prevent the timer from being reset.
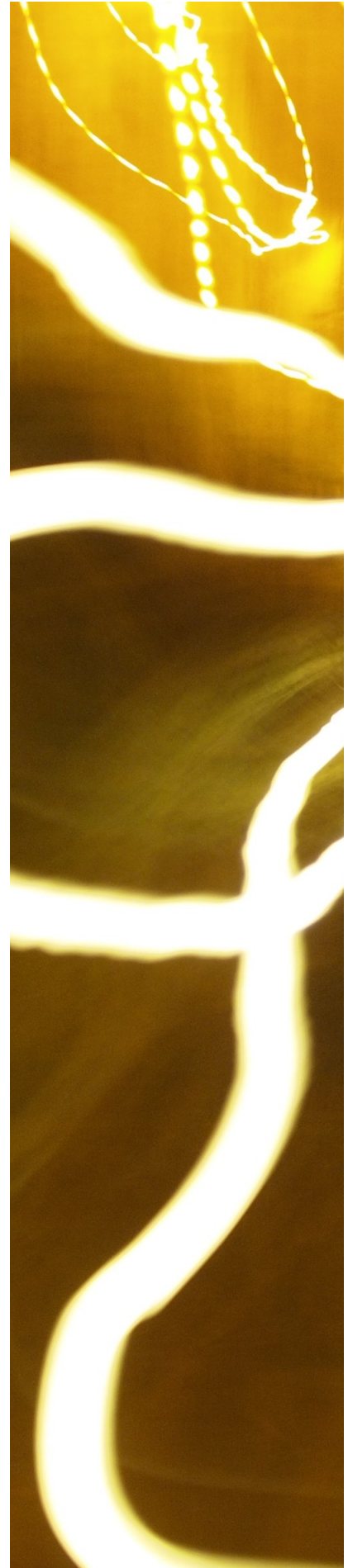
9 Null-pointer Checks

gcc deletes null-pointer checks beyond the first use of a pointer at optimization level two or higher

```
void bad_code(void *a) {
        int *b = a;
        int c = *b;
        static int d;

        if (b) {
                d = c;
        }
}
```

On the third line of code, we set c = *b. gcc assumes that this would trigger a hardware fault if b is zero, so therefor b cannot be zero.

This causes gcc to completley eliminate the check if (b) later in the file, so d = c is always run.

This can be seen in linux kernel 2.6.30

```
struct sock *sk = tun->sk; // sk initialized to tun->sk// ...
if (!tun) return POLLERR;
```

This code will never return POLLERR because it is assumed that tun cannot be zero. This could be locally exploted by mapping page zero with mmap() then triggering the bug in the process's context, creating an execution jump to attacker-controlled data.

This was fixed in kernel version 2.6.23 by moving the definition of sk down to below the if statement.

10 That's All, Folks

This has been a summary of Dangerous Optimizations and the Loss of Causality, a lecture given in 2010 by Professor Robert C. Seacord at Carnegie Mellon, CS 15-392.
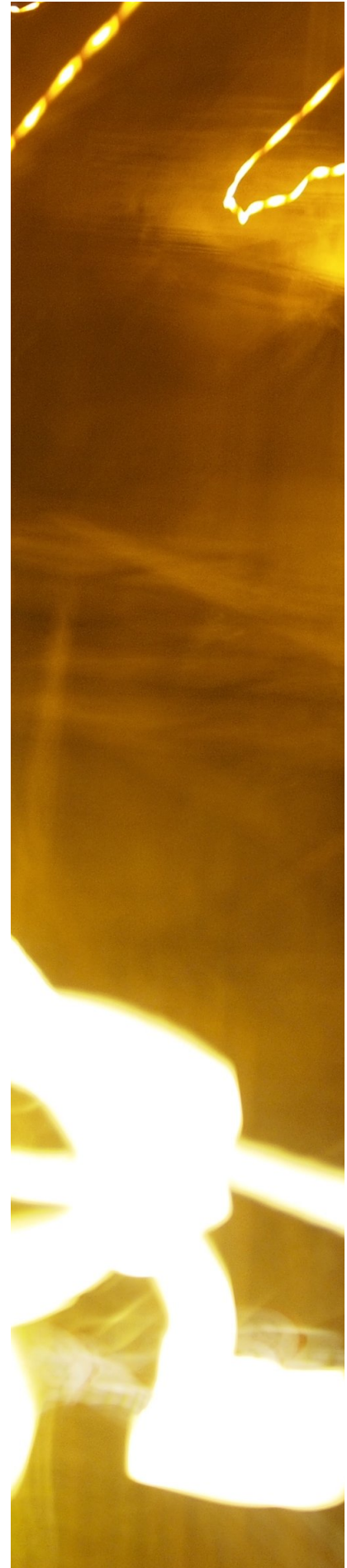
Electric Defense 2
By Victor

Electric weapons 101

In a world ruled by electricity, taking advantage of its power can determine the fate of an individual, either by using it to run computers and sophisticated devices in order to gain advantage over your adversaries or simply using its raw power in an offensive way. Nowadays, when nearly every device and home appliance has its own electric circuit, it doesn't take much skill to turn inoffensive circuits into dangerous electric weapons.

Electric weapons haven't changed much since they were devised, the main principle of operation behind all of them is to create an electric potential high enough so current can flow inside the body, overflowing the nervous system with electric signals in an extremely painful way, losing control over the muscles, causing them to contract and leaving the subject incapable of fighting back even after the discharge has ceased due to the muscles needing some time to recover.

These electric weapons have many advantages, since they usually aren't lethal, leave little to no evidence after having been used and just require a slight contact with the subject in order to do their job, a good tool to have to get out of a threatening situation without any further consequences.

Understanding how they work

One of the most important equations in electronics is voltage equals current times resistance. Now, for our electric weapon we need a lot of voltage, enough so electrons want to fly off the electrodes, enter the body of our attacker and be sucked again into the circuit, if we increase the voltage we'll decrease the output current, that means we'll need a considerable amount of input current to start with, 9v alkaline batteries are not the best choice, Ni-Mh batteries are better, but they can't come close to lithium batteries in terms of current delivery and energy density, but we would need at least three of them in series to get a nice input voltage to start with. For practical reasons I'll use a 9V Ni-Mh battery, although lithium battery packs are advised.

Now we need to find a way to turn those 9 volts into 10000 volts at least, that will give us around 1Cm spark, enough to jump through clothes directly into the body, although our final voltage will probably be around 15000 volts. One of the best and simplest ways to increase a voltage is to use magnetism, this is, temporarily storing electric energy into magnetic energy, only to be converted back again with a much higher potential, but lower current. We will rely on the basic laws of magnetism, which state any change in magnetic field passing through a loop of wire will create an electric potential, this potential is directly related to the number of turns in the loop, the higher this number is, the higher the voltage will be.

To put these laws in practice we'll use a flyback transformer. Flyback transformers consist of two coils, a primary and a secondary, wound

around a ferrite core. This ferrite core stores the magnetic field created by the primary and when it collapses it gets transformed into a higher voltage by the secondary. According to the magnetism laws mentioned before, the ratio of turns of copper wire must be very high, around 1:100 or higher, in order to produce high voltages. A 555 timer will provide the electric pulses needed to alternate the magnetic field.

To further amplify the voltage a Cockcroft-Walton multiplier is used at the output of the transformer, this simple circuit relies on capacitors, which store electric charges and allow AC current to pass, and diodes, which act like check valves, only allowing the charges to travel forward, in combination the potential is increased while the current diminishes at each stage.

Getting started

After the concepts are clear we can start building our electric weapon or stun gun. We'll start by gathering all our electronic components, an electronic "junk box" can help in some cases.

Parts needed

- Some broken CFL bulbs
- 555 timer chip
- 220-470uF 25volt capacitor
- 10uF 25 volt capacitor
- 1uF 25 volt capacitor
- 5.6nF capacitor (film or ceramic)
- Switch (flush preferred)
- Pushbutton
- Battery connector
- MOSFET of choice (IRF540)
- 10k Potentiometer
- 470R, 2x330R, 10R Resistors
- 20x UF4007 or BA159 diodes (or any fast recovery diode) you can also use 10x 2-3 kV rated
- 10x 1nF 3kV capacitors
- 2x Screws, nuts and locking washers
- Plastic box or case (non metallic)

Tools needed

- Soldering iron
- Drill

Construction

The flyback driver board is assembled first, follow the circuit diagram to assemble it, to avoid voltage spikes caused by the discharges of the stun gun make traces as short as possible, wire bridges if any should be flush with the board and must be as short as possible, use copper clad boards or prototype boards with appropriate trace width for high current sections. Build the multiplier on a different board. If you're unsure about how to connect the multiplier to the transformer at the driver board you can connect a neon bulb to the output of the transformer and power the driver, the electrode with a brighter glow around it will be the negative and it will be connected to the negative of the multiplier, the dimmer one will be the positive.

To make the flyback transformer, a transformer from a fairly big CFL PSU is desoldered, after removing the protective tape, the ferrite core is evenly heated up with a small torch until the glue holding it together melts, the two "E" parts of the core are extracted with the help of some pliers. Now the windings are exposed, snip the wire and begin pulling it out until there's nothing left but the plastic to hold a new winding, clean glue residues with alcohol. Tightly wind around 20 turns of ~24 AWG enameled wire to create the primary, for the secondary I used a very thin enameled wire, around 35 AWG as calculated according to its resistance per meter, wind as many turns as necessary to fill the transformer, probably around 1000 turns in this case. Solder the ends of each coil to a lead at the base of the transformer. To finish it all, dip the transformer in hot wax, waiting around half a minute so it can get between the windings. Some kind of cyanoacrylate glue could work too, although this isn't necessary it gives me more confidence, as in many occasions I could smell ozone being generated by the transformer, probably indicating a short.
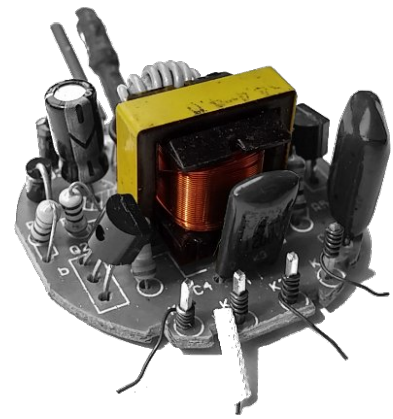
CFL power supply with desired transformer.

<50% Duty Cycle || 9.5–60kHz

Capacitors: 3kV 1nF || Diodes: Fast recovery 3kV
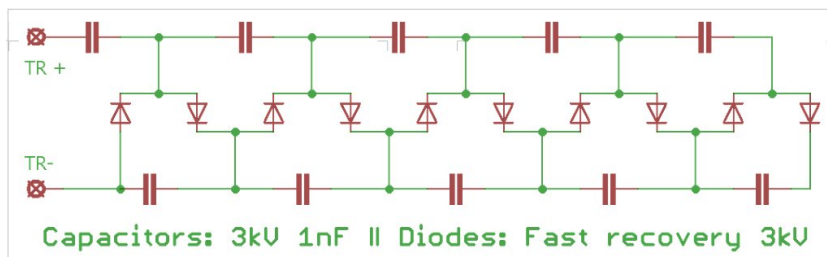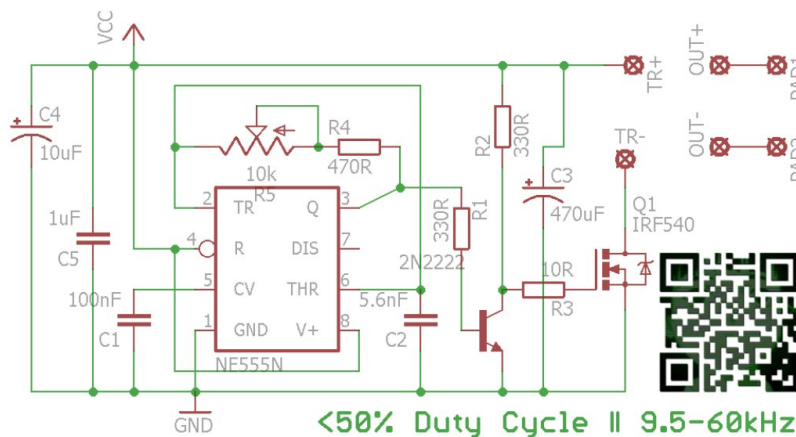
To make the multiplier a perforated board without pads can be used, assemble the circuit keeping clearances in mind, when soldering and trimming the leads avoid pointy ends, as electrons like to jump off them,

this effect is known as corona discharge. Leave at least 5mm between the capacitor leads, using a padded perfboard is not advised as the copper pads make it difficult to respect the clearances. In this case 1nF capacitors rated 3 kilovolts have been used in conjunction with BA159 diodes, as it can be seen, 20 diodes have been used, this is because the output voltage of the transformer can get up to 2 kilovolts, instead of ordering 2kv rated diodes, 2 diodes in series capable of 1kv reverse voltage each will work just fine.

Laying the circuit inside the case is also important, try to maintain a distance between the high voltage multiplier and low voltage components. Holes for the switch and pushbutton are made in convenient locations, they are placed and soldered to the circuit, use plenty of shrink tube and electrical tape in case of doubt. Double sided tape or hot glue can be used to hold the boards in place and packing foam to separate them. To make the output contacts two holes are made through one extreme and two bolts are inserted with their nuts and locking washers, clamping the exposed output wires in between, some sheet metal from a tin can is used to make the spark gap, which in this case is 1.3Cm apart, experimenting with different configurations is encouraged, as arcing is kind of unpredictable. As an optional improvement, a capacitor can be connected between the output terminals, this will create beefier sparks that will pack a bigger punch, the capacitors should be rated for 15 to 20kV, this can make them harder to find. Their capacitance value will vary depending on the space left in each case, as these capacitors tend to be quite big.

The trigger will be a pushbutton that will connect the battery to the 555 and flyback transformer, avoid using pin 4 of the 555 as the trigger, as the voltage spikes will affect the performance of the circuit. As a measure of safety to avoid accidentally turning on the device, a switch is installed. Both the switch and the pushbutton should be rated for at least 3 amps.

To adjust the stun gun open the case and hold it into the air, as leaving it on a desk or any other surface while performing this operation would include parasitic capacitances that will affect the correct adjustment, with the pushbutton constantly pressed adjust the potentiometer with a screwdriver until the rate at which sparks occur is higher. Adjust the spark gap to the widest possible and repeat the operation until it can't be further optimized.

Electric noise: a stun gun is probably one of the worst environments for a circuit to work, high current and high voltage pulses produced by the arc and the transformer induce voltage spikes inside the circuit, a couple of capacitors between power and ground to filter these spikes is fundamental but this doesn't completely solves the issue, insulating the 555 driver using a faraday cage made with conductive paint or HVAC foil tape could be a simple solution.

Other uses: The electric disturbances produced by this device can be strong enough to alter the normal operation of other electronic devices or even destroy them, operating the stun gun at a distance closer than 2cms from a circuit board will probably crash or brick it, although at this point one might just zap it if the intention is to destroy it, adding a coil to the output of the stun gun with a small spark gap will amplify this effect, allowing you to mess with circuit boards locked behind a plastic casing, this won't work with metal cases as they will absorb the energy produced by the magnetic field before it reaches the circuit.

Other ways of making a stun gun: One of the most used stun gun circuits uses a pulse trigger transformer, these transformers are hard to make and require isolating the coils with resin and purging the air to avoid shorts at the secondary. Both circuits are similar, the difference being

the stun gun described here converts the voltage coming out of the first transformer using a Cockcroft-Walton multiplier, which is far easier to make from common components, while the standard one stores the output in a capacitor to discharge it though this second transformer, further amplifying it.

This same 555 driver circuit can be used without many modifications with bigger transformers, for example, to make electric fences.

Troubleshooting

- It just doesn't works:

Check the battery is fresh, recheck connections, make sure the 555 timer works.

- 555 and MOSFET are working but there's no output.

Check the transformer is connected, check the output of the transformer with a neon bulb, if it doesn't glows despite current flowing through the primary it is shorted or there is a problem with it. Check the transformer has been connected properly to the multiplier, pay attention to the polarity.

- MOSFET/Transformer are overheating

Very low primary impedance, increase turns at the primary or increase frequency. Check gate of the MOSFET, there should be a square wave there.

- It was working but now the output has dropped a lot.

Check diodes on the multiplier with an ohmmeter in the megaohms range, any diode with low or different reverse values than a new one is fried. Capacitors are not likely to fail, but they can be tested too. If diodes fry often, the output of the transformer is too high, decrease the number of turns at the secondary or increase the rating of the diodes/put another diode in series. If you smell ozone/burning near the transformer it might be shorted. Test it in the dark to check for unwanted arcing or corona discharge.

(1) Switch
(2) Pushbutton
(3) Battery
(4) 555 driver
(5) Adjust potentiometer
(6) Flyback ransformer
(7) MOSFET or high power transistor
(8) Main capacitor
(8) Main capacitor (470uF)
(9) Noise filter capacitors (10 and 1uF)
(10) Multiplier board
(11) Spark gap
(12) Output terminals
(13) Rubber isolation
(14) Optional output capacitor

12x5.5x3cm
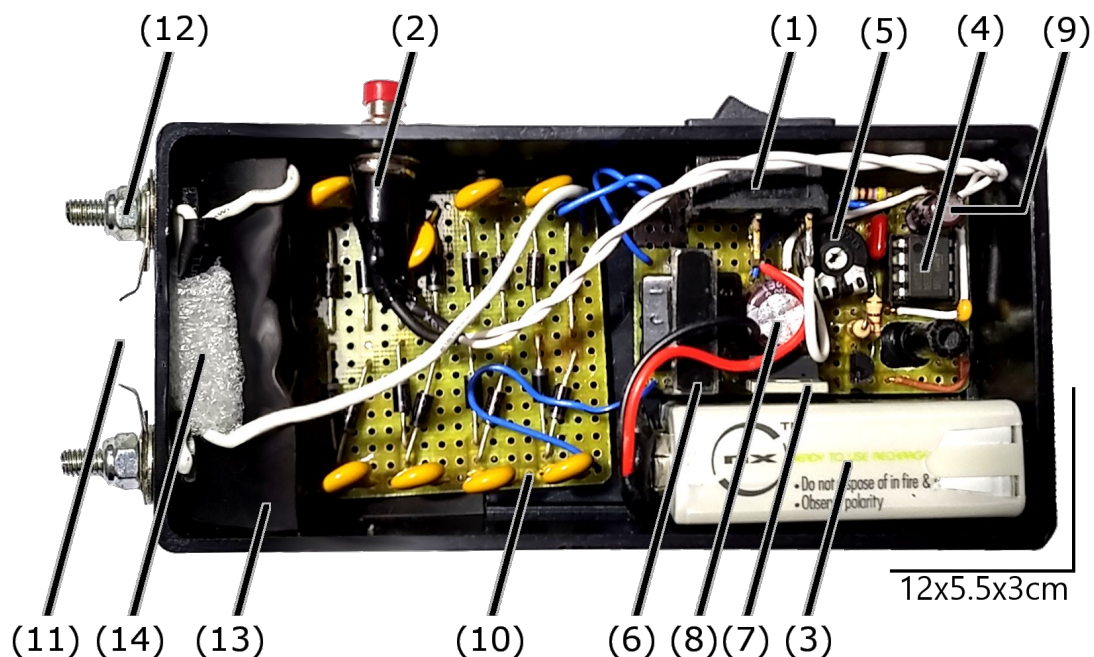
EM Spectrum
By JS

Electro Magnetic Spectrum

Wave

10 Mm	1 Mm	100 km	10 km	1 km	100 m	10 m	1 m	10 cm	1 cm	1 mm	100 μm	10 μm
10^7 m	10^6 m	10^5 m	10^4 m	10^3 m	10^2 m	10^1 m	1 m	10^{-1} m	10^{-2} m	10^{-3} m	10^{-4} m	10^{-5} m

Radio

LW	MW	SW	VHF (30 MHz - 300 MHz)	UHF (300 MHz - 3 GHz)

Microwave

SHF (3 GHz - 30 GHz)	EHF (30 GHz - 300 GHz)

Infrared

Radar

(3 MHz - 30 MHz)	(50 MHz - 300 MHz)	(300 MHz - 1 GHz)	(1 GHz - 2 GHz)	(2 GHz - 4 GHz)	(4 GHz - 8 GHz)	(8 - 12)	(12 - 18)	(18 - 27)	(27 - 40)
HF over the horizon	VHF very long range	UHF long range	L long range air traffic	S long range weather	C marine weather	X radar	K_u satellite	K speed camera	K_a mapping altimetry

A pigment is a material coloured by absorbing selective frequences of light, they are insoluble. Dyes and inks contain pigments (by bonding with them at a molecular level) and are dissolved in a solvent. Dyes have an affinity to the substance they are applied. Inks remain on the surface.

Pigments work by absorbing photons of selective frequencies, allowing the rest to be reflected. The energy of these photons is absorbed by the nucleus of the atom as heat. This is why white materials remain cooler in the sun than black ones, black absorbs lots of photons where as the white reflects most of the photons. This selective absorption results in a change of colour as most of the original colour used to light the surface is not reflected; Hence white light will be reflected as red because all of the yellow, green and blue has been absorbed, leaving only the red light to be reflected back at the viewer.
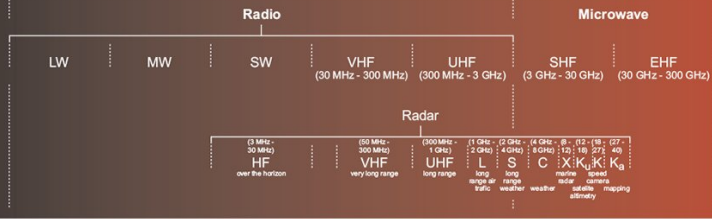
In a flourescent pigment the incoming photon is absorbed by an electron in the pigment. This energy is used to temporarily shift the electron from a low energy state to a high energy state. As the electron then naturally returns to the lower energy state it emmitts a photon of a different (lower) frequency to that of the photon originally absorbed. This is how U.V. ink works on bank notes. When you hold it up to a high frequency (U.V.) light you can see the ink glow bright blue. You do not see the ink reflecting U.V. light, rather you see it re-emmiting a lower than U.V. frequency, blue visible light. This process is called down-conversion.

Up-conversion is a similar process but instead of converting a high energy photon to a low energy photon, it converts two (or more) low energy photons into a single high energy photon. This requires high levels of illumination input in order to get low levels of output.

(N^2O)
(CH^4)
$(O + O^3)$
(CO^2)
(H^2O)

10 μm
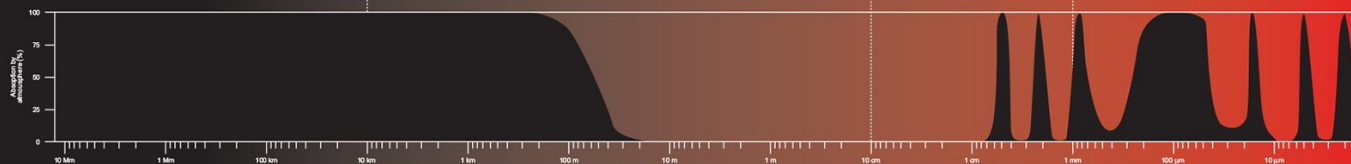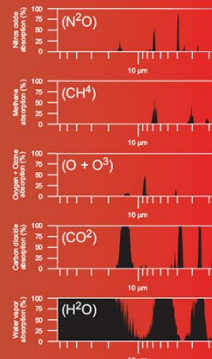
Ionosphere

Ionosperic reflection of low frequency radio waves does not only reflect waves back from outer space, it also reflects back low frequency radio waves from the surface. This effect alows LW radio to bounce around the world.

mesosphere
ozone
stratosphere
troposphere

Ionosphere reflection   Radio window CO^2, H^2O, O, N^2O, NO^5

1 K°   100 K°   10

Long wavelength, low energy

Rayleigh scattering

Total

210-310 K

All electromagnetic radiation (everything on this chart) is photons. The only diffrence between radio waves, light waves and X-rays is the energy they contain and hence the frequencys at which they vibrate.

Satelite T.V. (10 GHz - 14 GHz)

3G upload (1.92 GHz - 1.98 GHz)
GPS download (1.575 GHz) (2.11 GHz - 2.17 GHz)

R.C. models (26.995 - 27.255 MHz) (50.8 MHz - 50.98 MHz) (72.01 MHz - 72.99 MHz) (75.41 MHz - 75.99 MHz)

GSM-900 upload (880 MHz - 915 MHz) download (925 MHz - 960 MHz)

GSM-1800 upload (1.71 GHz - 1.785 GHz) download (1.805 GHz - 1.88 GHz)

UK T.V. (474 MHz - 850 MHz)   Microwave oven (2.45 GHz)

Outgoing thermal radiation 15 - 30% transmitted

AM LW (148.5 kHz - 283.5 kHz)	AM MW (520 kHz - 1.6 MHz)	AM SW (2.3 MHz - 26.1 MHz)	FM (87.9 MHz - 107.9 MHz)	WiFi 802.11 g/b (2.4 GHz)	WiFi 802.11 a/n (5 GHz)

Thermal vision (22 THz - 32 THz)   Night (30

Millimeter   Sub-millimeter

100 Hz	1 kHz	10 kHz	100 kHz	1 MHz	10 MHz	100 MHz	1 GHz	10 GHz	100 GHz	1 THz	10 THz	100 TH
10^2 Hz	10^3 Hz	10^4 Hz	10^5 Hz	10^6 Hz	10^7 Hz	10^8 Hz	10^9 Hz	10^{10} Hz	10^{11} Hz	10^{12} Hz	10^{13} Hz	10^{14} Hz

Frequ

100 feV	1 peV	10 peV	100 peV	1 neV	10 neV	100 neV	1 μeV	10 μeV	100 μeV	1 meV	10 meV	100 meV
10^{-13} eV	10^{-12} eV	10^{-11} eV	10^{-10} eV	10^{-9} eV	10^{-8} eV	10^{-7} eV	10^{-6} eV	10^{-5} eV	10^{-4} eV	10^{-3} eV	10^{-2} eV	10^{-1} eV

Electr

Diagram 2 from a series of x. Resurch and layout: JS

length λ (m)

1 μm	100 nm	10 nm	1 nm	100 pm	10 pm	1 pm	100 fm	10 fm	1 fm	100 am	10 am	1 am	100 zm
10^{-6} m	10^{-7} m	10^{-8} m	10^{-9} m	10^{-10} m	10^{-11} m	10^{-12} m	10^{-13} m	10^{-14} m	10^{-15} m	10^{-16} m	10^{-17} m	10^{-18} m	10^{-19} m

Visible   **Ultraviolet**   **X - rays** photons generated by energetic electrons   **Gamma rays** photons generated by transitions in atomic nuclei

750 nm   380 nm

"hard" X-ray and Gamma ray frequencys overlap.
Definition is given by process of creation

Krypton (2200 °K)
Neon (2700 °K)
High pressure sodium (3000 °K)
Incandecent bulb (3200 °K)
Halogen (4100 °K)
LED - "soft white" (4500 °K)
Fluorescent - "cool white" (5600 °K)
LED - "cool white" (6500 °K)
Xenon (12000 °K)
Argon (20000 °K)
Mercury (22000 °K)

Radiant energy

Human eye (long-wavelength cone)
Human eye (mid-wavelength cone)
Human eye (short-wavelength cone)
Human eye (Rods)
Typical CCD
Typical CMOS
Photoresistor - LDR (cadmium sulfide - CdS)
Silicon phototransistor

Relative sensitivity

Wavelength (nm)

Resurch, GRFX, layout by JS – pre-release version for the LainZine.

Optical window   O³   O, N

Temperature required of a black body to emmitt photos of coresponding frequency

100°K°   10000 K°   10000000 K°

Short wavelength, high energy

All of the colours which you perceive are constructed from the relative intensities of three wavelength areas: red, green and blue (long wavelength cone, medium wavelength cone and short wavelength cone).
It is combinations of just these three signals that renders our perception of the full range of colours we see. For example: Dark red light will only excite the long wavelength cone. Bright red light will excite the long wavelength cone and also slightly excite the medium wavelength cone. Green light will excite the long wavelength cone slightly, the medium wavelength cone considerably and the short wavelength cone modretly. Blue light will not excite the long wavelength cone, excite the medium wavelength slightly and the short wavelength cone will be heavily excited. By reading the mixed intensity of these three signals your vision can distinguish over 100000 different huws.

5525 K

Incoming solar radiation
70 - 75% transmitted

Cobalt-60
food irradiation
(1.17 MeV - 1.33 MeV)

Medical "soft" X-ray
used for imaging
(30 PHz - 6.5 EHz)

Medical "hard" X-ray
used for radiotherapy
(6.5 EHz - 30 EHz)

Caesium-137
blood treatment
(662 keV)

Uranium 235
fission fuel
(4.5 MeV)

Cosmic rays
energies >10²⁰ eV have been measured!
(100 MeV - 10 GeV)

SLAC
particle accelerator
(50 GeV)

CERN LHC
particle accelerator
(450 GeV - 7 TeV)

vision
0 THz)

1 PHz	10 PHz	100 PHz	1 EHz	10 EHz	100 EHz	1 ZHz	10 ZHz	100 ZHz	1 YHz	10 YHz	100 YHz	1000 YHz
10^{15} Hz	10^{16} Hz	10^{17} Hz	10^{18} Hz	10^{19} Hz	10^{20} Hz	10^{21} Hz	10^{22} Hz	10^{23} Hz	10^{24} Hz	10^{25} Hz	10^{26} Hz	10^{27} Hz

quency (Hz)

1 eV	10 eV	100 eV	1 keV	10 keV	100 keV	1 MeV	10 MeV	100 MeV	1 GeV	10 GeV	100 GeV	1 TeV	10 TeV
1 eV	10^1 eV	10^2 eV	10^3 eV	10^4 eV	10^5 eV	10^6 eV	10^7 eV	10^8 eV	10^9 eV	10^{10} eV	10^{11} eV	10^{12} eV	10^{13} eV

ron volt (eV)

Hunter Eat Hunter

By nullmuse

■ What story should I tell you, lainon? A true one:

chattr +i ./*

And then I begin my collection. I can imagine his face, see the glint of cold sweat breaking out on his brow. He moves to stop me, to attempt damage control.

rm: cannot remove 'svchost.exe': operation not permitted
rm: cannot remove 'xiao': operation not permitted
rm: cannot remove 'xynsyn': operation not permitted

On and on. My scp command continues, pulling their entire toolkit home with me. I watch carefully, spamming processlist checks in a separate window. *What is he doing? Where will his panic lead him next? Will he disconnect me before I grab every RAT and exploit in their bag?*
He doesn't. I get away with everything. I fire off a recursive sed, bestowing amnesia upon every log file that ever heard of my IP address, and then kill -9 $$. My trace disappears, and I return home with my cargo.

I run honeypots for a hobby. Sometimes I hack back. This time around a target from China was setting up one of my pots as an exploit kit server. I watched carefully, monitoring his actions closely. I often assume the persona of the trapdoor spider – sensing the vibrations of a cricket mere inches above me. Monitoring, waiting.

Upgrading my server. Installing nginix. Modifying PHP scripts.

Then he creates a backdoor. A useradd command flashes past my terminal. A password.
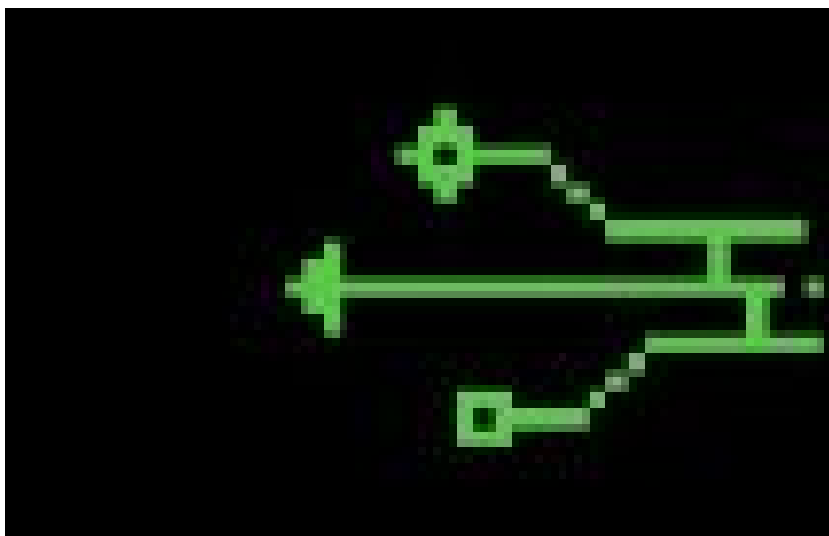
A thought passes: *Is this stranger stupid enough to reuse passwords?* I connect to a Ukrainian hop point, and then ssh to the intruder.

user: root
password: hu@ng!!23

A pound sign greets me and I laugh. id. uname -a. ps -eaf. netstat -tunalp. ls -la.

A massive directory structure rises to meet me. Dozens of tools scroll past my screen -- RATs, exploits, trojans, rootkits. Linux, Windows, Mac, Mikrotik. Food for reversing. A veritable goldmine for new techniques and adversary tactics.

id. Someone else is here.

I wonder what I would do in his situation, if he saw me? Delete everything. Disconnect the attacker. I cannot control the later, so I recursively immortalize everything I see, and start grabbing. My adversary panics, a wonderful emotion, and I get away.

So here I am, tearing through Chinese malware with edb, vivisect, and bokken. These guys are good. Listening Post obfuscation, only discoverable through late nights staring at assembly. One RAT silences itself the moment Wireshark enters the process list. Rootkits modifying kmem. Flash exploits. Unknown ones.

I run honeypots. I like to catch the bad guys. But I don't do it for the good guys. I just like to stay current.

IRC bot
By Gitgood

Writing an IRC bot in Python (3.5)

May 6, 2016

Abstract

This article will hopefully introduce the reader to IRC, basic sockets programming in Python and general IRC bot creation. IRC has been an incredibly popular and prominent mode of communication in the cyberpunk/programming/tech literate communities from the early 90's, and because of this many great communities have and continue to flourish and mature on these networks. Because of the simplistic nature of IRC, the creation of "bots" that can provide useful functions in IRC channels is surprisingly trivial and easy to pick up. Also, I am aware the code formatting is less than perfect so if you just want to grab the code, here's a link: http://pastie.org/10826422

### What is IRC? Why would you need a bot?

What is IRC?

Developed in late 1988, IRC stands for "Internet Relay Chat" and is a massively popular communication platform.

IRC lets users connect to a server and communicate with other connected users (usually through some IRC client) in real time, even allowing the transfer of files. IRCP (Internet Relay Chat Protocol) is the application layer protocol which helps facilitate the communication of these IRC clients.

When connected to an IRC server, a user can issue various different commands. A few examples of these are:

Command	What is does
/away MESSAGE	Leaves a message explaining why you're away.
/help	Displays a list of all commands.
/join CHANNEL	Joins #channel
/message USER	Sends message to USER
/nick NICKNAME	Sets user's nickname to NICKNAME[6]

Once connected to a server, a user still needs to join a channel before they can begin chatting. Channels can be described as chatrooms that people can join and leave at will. So, for example, if a user wants to connect to freenode's #python channel they would first join irc.freenode.net using their IRC client and after all the MOTD (Message of the day) text would type /join #python. From here the user is free to chat away to the other users connected to #python. IRC gained massive popularity in the 90's in the programming/hacking/technology-literate subcultures. "In the autumn year 2000, EFnet has some 50,000 users and IRCnet 70,000."[5] As it stands today, the largest IRC networks currently are:
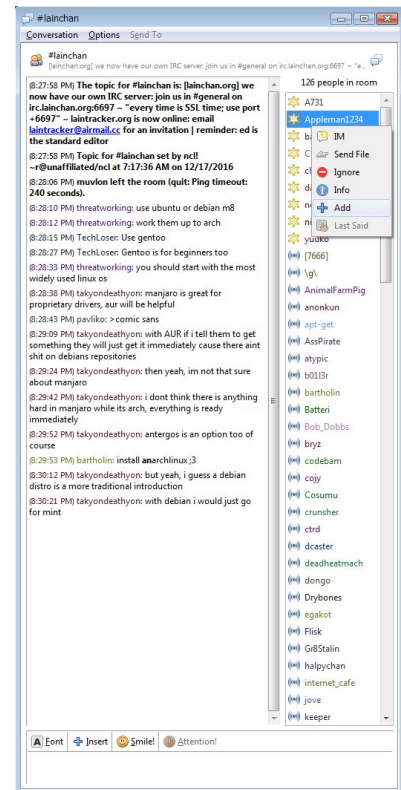
Lainchan IRC

Total IRC abuse.

- IRCnet - 38716 users.
- QuakeNet - 34879 users.
- EFnet - 22790 users.
- Rizon - 21030 users.[2]

Because of the massive popularity of IRC, if you have an interest then chances are there is a community for it.

Bots? Why?

IRC itself is fairly simple and as a result doesn't come jam packed with features. People found that due to the simplistic nature of IRC, computer programs could easily be written to provide certain features that IRC lacks. For example, a bot could be written to emulate a magic 8-ball. It would listen on a channel for a command such as "!8ball [QUESTION]", and once found would choose a random 8-ball phrase and send it back to the chat. It would look something like:

```
<gitgood>    !8ball will I die soon?
<8ballbot>   Without a doubt.
<gitgood>    damn.
```

This, of course, is a very simple example but you get the gist. This is the bot we are going to be writing.

Writing the IRC bot

Now that the basics of IRC is out of the way, we can now get to implementing the bot.

Setting up the bot

As mentioned previously, we're going to be writing this magic 8ball bot in Python 3.5 so some Python experience is suggested. Firstly, we need to import two modules. These are "socket" and "random". The socket module will provide us with network communication, and the random module will be used to select a random magic 8ball phrase. If you don't know much about sockets, I suggest reading:

- import socket
- import random

After this we're going to need the magic 8-ball phrases in a list. Then, when the time comes, we can choose a random phrase from the list. Magic 8-balls typically have 20 standard phrases. 10 of these phrases are positive (Yes), 5 of these are neutral (Ask again later) and the remaining 5 are negative (My sources say no). Here is the finished list:

```
phrases = [
    "It  is certain.", #First ten phrases are "positive"
    "It is decidedly so.",
    "Without a doubt.",
    "Yes, definitely.",
    "You may rely on it.",
    "As I see it, yes.",
    "Most likely.",
    "Outlook good.",
    "Yes.",
    "Signs point to yes.",
    "Reply hazy, try again.", #Five "neutral" phrases.
    "Ask again later.",
    "Better not tell you now.",
    "Cannot predict now.",
    "Concentrate and ask again.",
    "Don't count on it.", #Five "negative" phrases.
    "My reply is no.",
    "My sources say no.",
    "Outlook not so good.",
    "Very doubtful."
]
```

Now that the phrases array has been made, we need to create some variables to store the IRC server, channel and bot nickname.

```
server = "irc.freenode.net"
channel = "#8ballbottest"
nickname = "magicbottest"
```

The next thing that needs to be done is the creation of the socket instance, the connection to the IRC server and the sending of some initial information.

```
irc = socket.socket(socket.AF_INET, socket.SOCK_STREAM).
```

```
irc.connect((server, 6667))
irc.send(("USER " + nickname + " " + nickname + " " + nickname + " :Magicbot\r\n").encode(encoding="UTF-8"))
irc.send(("NICK " + nickname +"\r\n").encode(encoding="UTF-8"))
irc.send(("JOIN " + channel +"\r\n").encode(encoding="UTF-8"))
```

The first line creates a socket instance with two parameters. The first parameter is socket.AF_INET, and this refers to the IPv4 address family. The second parameter is socket.SOCK_STREAM, and this "Provides sequenced, reliable, two-way, connection-based byte streams."[3]

The next line uses the socket to connect to the freenode server on port 6667. "The well known TCP port for IRC traffic is 6667"[4]

After connecting to the server there are three lines that are for sending data to the server. The first line sends the USER command in the format:
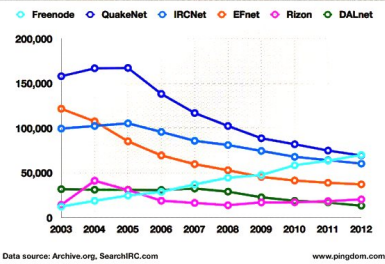
USER username hostname servername :realname

IRC bots activity - MSRT 12 months trending
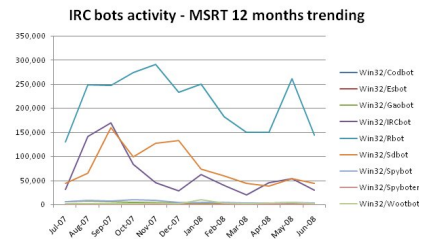
"No one could have anticipated all the ways that Internet Relay Chat (IRC) would eventually be used when it was 'created' in Finland during the late 1980s. People really started picking up on IRC in the early 1990s."

Top 6 IRC networks: number of users

○ Freenode ○ QuakeNet ○ IRCNet ○ EFnet ○ Rizon ○ DALnet

Data source: Archive.org, SearchIRC.com   www.pingdom.com

IRC Usage

Peaking in 2005 with a total of around 450,000 users across the top 6 IRC networks, falling to approximately 300,000 by 2012. Freenode being the only network enjoying a sustained increase in user base.

"The username is the user part in your user@host hostmask that appears on IRC, which shows where your connection comes from. The realname is used to populate the real name field that appears when someone uses the WHOIS command on your nick."[1]

Making the bot respond

Now that the bot has connected to the IRC server and joined the channel, it now needs a way of getting information from the channel and parsing that for use. Fortunately, this is very easy too.

```python
while True:
        #Recieve data from the socket, and decode it.
        recieved = irc.recv(2048).decode("UTF-8")
        print(recieved)
        #print(bytes(recieved, "UTF-8"))
        #If the server sends a PING.
        if recieved.startswith("PING"):
                #Respond with a PONG to prevent timing out.
                irc.send(("PONG " + recieved.split()[1] + "\r\n").encode())
                print("Ponged")

        if ":!8ball" in recieved:
                #Splits the recieved response in to a list with two
elements. [0] is what was before the !8ball, and [1] is what was after.
                #It then gets what was after "!8ball", and calls .strip()
on this string to remove any trailing whitespace characters
                #such as "\r" and "\n"
                question = recieved.split(":!8ball")[1].strip()
                #If there has been a statement after !8ball such as
"!8ball am I going to die?"
                if question != "":
                        #Then send a a random phrase from the
phrases array to the channel.
                        irc.send(("PRIVMSG " + channel + " :" +
random.choice(phrases) + " \r\n").encode())
```

Everything is in an infinite loop as we always want to be recieving data and parsing it. We recieve at most 2048 bytes from the connection, and then decode it to UTF-8. After decoding the recieved data we then print it. Errors may occur here if the channel the IRC bot is joining has Unicode characters in it. If so, just uncomment the next line instead. To ensure that the client hasn't timed out, the server will ocassionally "PING" the client. When a message that starts with "PING" is found, then we send back the appropriate "PONG" to let the server know we're still connected. If "!8ball" is in the recieved data, then we can assume someone has tried to summon the magic 8ball bot. We then get the text after "!8ball". If the question is equal to "", then no question has been asked after "!8ball" and therefor no response should be given. If there is a message after "!8ball", then send a random choice from the phrases array to the IRC channel.

Complete code:

```python
import socket
import random

phrases = [
    "It is certain.", #First ten phrases are "positive"
    "It is decidedly so.",
    "Without a doubt.",
    "Yes, definitely.",
    "You may rely on it.",
    "As I see it, yes.",
    "Most likely.",
    "Outlook good.",
    "Yes.",
    "Signs point to yes.",
    "Reply hazy, try again.", #Five "neutral" phrases.
    "Ask again later.",
    "Better not tell you now.",
    "Cannot predict now.",
    "Concentrate and ask again.",
    "Don't count on it.", #Five "negative" phrases.
    "My reply is no.",
    "My sources say no.",
    "Outlook not so good.",
    "Very doubtful."
]

server = "irc.freenode.net" #We"re connecting to the freenode IRC server.
channel = "#8ballbottest" #A (probably) empty channel for testing the bot.
nickname = "magicbottest" #The nickname of the bot.

irc = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #Create
a socket instance.

irc.connect((server, 6667))
irc.send(("USER " + nickname + " " + nickname + " " + nickname + "
:Magicbot\r\n").encode(encoding="UTF-8"))
irc.send(("NICK " + nickname +"\r\n").encode(encoding="UTF-8"))
irc.send(("JOIN " + channel +"\r\n").encode(encoding="UTF-8"))

#Infinite loop.
while True:
    #Recieve data from the socket, and decode it.
    recieved = irc.recv(2048).decode("UTF-8")
```

References

[1] Caf. What is the difference between the nick, user-name, and real name in irc, and what is the password? http://stackoverflow.com/questions/31666247/what-is-the-difference-between-the-nick-username-and-real-name-in-irc-and-wha, 2016.

[2] A. Gelhausen. Irc networks - top 100. http://irc.netsplit.de/networks/top100.php, 2016.

[3] M. Kerrisk. Socket(2) - linux programmer's manual. http://man7.org/linux/man-pages/man2/socket.2.html, 2015.

```python
    print(recieved)
    #print(bytes(recieved, "UTF-8"))
    #If the server sends a PING.
    if recieved.startswith("PING"):
        #Respond with a PONG to prevent timing out.
        irc.send(("PONG " + recieved.split()[1] + "\r\n").encode())
        print("Ponged")


    if ":!8ball" in recieved:
        #Splits the recieved response in to a list with two elements. [0]
is what was before the !8ball, and [1] is what was after.
        #It then gets what was after "!8ball", and calls .strip() on this
string to remove any trailing whitespace characters
        #such as "\r" and "\n"
        question = recieved.split(":!8ball")[1].strip()
        #If there has been a statement after !8ball such as "!8ball am I
going to die?"
        if question != "":
            #Then send a a random phrase from the phrases array to the channel.
                        irc.send(("PRIVMSG " + channel + " :" +
random.choice(phrases) + " \r\n").encode())
```

How else can they be used?

I'm glad you asked! One very malicious use for IRC bots is the commanding of *botnets*. As you probably know,

botnets are a large network of infected "zombie" computers that can be controlled (often maliciously). For instance,

say you've a network of 10000 bots, and you command them to spam a webhost with requests then that could possibly take

that server offline (depending on how large they are).

How would you command this server? You could directly send the command to each and every "zombie", but that doesn't seem very efficient. What you *could* do however, is include some IRC bot code not unlike the code we wrote previously so that when a computer is infected the virus automatically connects to a certain server/channel. Once it has connected, it can then listen for commands. An example might be:

    <1337man> !ddos https://www.volafile.io

Each one of these infected computers would see this command, and then execute some function that would flood the link with requests.

[4] L. MikeDuigou. Internet relay chat (irc). https://wiki.wireshark.org/IRC, 2008.

[5] D. Stenberg. History of irc (internet relay chat). https://daniel.haxx.se/irchistory.html, 2011.

[6] Unknown. Irc information.... http://www.ircbeginner.com/ircinfo/ircc-commands.html, 2013.

Japan Mainland
By Tom Millicent

Japan

As a part of the replication process I was given access to this information.

I was getting so tired of having to fill those boxes ~ there was no way I'd meet this month's quota. I'd run into a bit of a situation with my teacher, we had an assignment due that required us to submit a working replica of a system found in society; simple I know!

The problem arose when my project had reached its final phase, I was the Japanese version and should have had no trouble with putting the pieces of my model bridge together - a functional suspension-bridge - turned out more tricky than I'd expected... the cars lined up on either side waiting for the gate to open, allowing water traffic to pass and visa versa. The boats weren't confined to lanes and took a staggered approach, whereas cars would line up behind one another trailing beyond the model.

I shared my design with our teacher before completing a model version. He had encouraged me to stick with my idea no matter how difficult it may seem. I'd successfully demonstrated that I listened intently in class, engaging with the topics that were set; this was no issue, that was one box filled. Upon hearing that my mainland counterpart would be joining us though, I began to behave strangely.

The supervisor at the school entrance mentioned my tally to me - you're a bit short this term, why don't you speak with your co-ordinator before you leave today? I'd happily done so ~ my friend waited for me after class.

"*What did he tell you?*" He asked.
"*He said that I wasn't filling enough boxes, that they're sending my mainland version to the school so I have to look after him.*"
"*Oh, I have not met a mainland counterpart before.*"

We all gathered in the recreation room that evening. Some kids were playing games, I was too exhausted from hearing that news so I watched television. There wasn't much of interest on; advertisements for retirement packages in serene locations in between teasers of the movie that was showing this weekend. I eventually retired to my room as an attempt at study - I had a lot of work left on my model, the bridge wasn't yet lowering; vehicles waited patiently in both directions, boats bobbing gently on the waters surface had right of way while the gate was opened. I only had to let the cars through and it would be a functioning suspension replica.

The next morning I arrived for class a little early as I was going to meet my self from the mainland. He was waiting at the entrance ~ smiling as I walked in his direction.

"*Good day, I'm from the mainland!*"
"*Morning, we should go inside. Class begins soon.*"

Mainland

I had hardly a recollection of my home, it appeared so far away. The school was wonderful ~ bright, wide hallways with lots of room for students led to classrooms that filtered sunlight through great windows. Today we were presenting our models so all the desks had been pushed up against the walls with an island in the middle for whoever was showing their work, dioramas and contraptions spread about the room ready to be brought into the middle.

When it was time for showing the suspension bridge, we both went up. I let my self do the talking - he was better with Japanese and had done most the work, so I couldn't do more than offer my presence. He demonstrated how the bridge would allow cars to cross from one side to the other: once the boats were ready to pass through that section of the river the gate would open, stopping all the traffic.

I looked toward the door and saw a clerk peering through a gap at me. He had a somber expression as though a smile was not possible; though much deeper down he wanted to express this sentiment, his scrutiny would be undermined.

Watching the model again we were nearly done, some boxes were left unchecked - I could see my friend was getting nervous. He lowered the gate allowing the cars to drive from one side to the other once more. The boats continued along the waterway...

We met for lunch near the eatery, my friend was not happy with his performance.

"I just don't understand what they want from me."
"Don't worry, I'm sure you'll do alright."

Just as we were about to leave, a teacher could be seen from the hall as if he was approaching us. Another clerk pulled him away through the swinging doors, his outcry muffled as he disappeared down the hall.

The next day we left the dormitory together. I had been experiencing a continuous buzzing in my left ear while my friend seemed more estranged than the day before. When we got to the school I couldn't help but feel I was being watched; my friend took my hand and led me away from class.

"We can't go in there!"
"What do you mean? We don't want to be late."
"They're watching, they know who you are and they will come for me. You have to follow me."

He ran along the great hallway, pulling me behind him. I could hear footsteps in the distance as we reached the entrance gate.

"Going somewhere?" The supervisor asked.

We ran past the gate to a small opening in the building's exterior wall, my friend began crawling through. I didn't look back though could hear the clerks closing in on us - I followed him through a service shaft that opened onto a rooftop. He was scrambling up a large vent; I ran after him, finding my way to the top where he sat, gazing mindlessly at the view - vast valleys stretched into the distance, it was like he hadn't been outside before.

I sat beside him. A television set was attached to the surface of the vent, pointed towards me. An image surfaced from the static ~ a clerk spoke to me.

"*You will remain where you are, don't move. We are coming to get you, it is not safe out there!*"

I kicked the monitor from the vent and it fell some distance before crushing another clerk who was approaching.
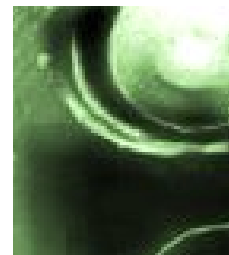
"*They are always watching us!*" I screamed at my friend.

He nodded and gave me a smile.

Practical LSA Extraction and Use: An In Depth Guide

By InsomniaLost

Disclaimer

Introduction

Hello, my name is Czech and I am a Chemistry Enthusiast with many, many years of experience both in academia and in the psychedelic world. I too, love lain. Lysergic Acid Amide AKA. Ergine is an alkaloid found in many plants, the list of which can be found below:

- Morning Glory
- Baby Hawaiian Woodrose Seeds
- Sleepy Grass
- Ololiúqui

Other plants I can not endorse due to potential for other compounds being present. LSA has been used for hundreds of years and is said to have first been discovered by Ancient Aztec agricultural scientists studying potential sustenance crops. The history is not so much what I cover here, as you could infer from the title, I am a chemist not a historian. It has low potential for physical side effects (at least in pure form).

Morning glory flower – Ipomoea nil.

Medical Disclaimer

There are 2 problems you are liable to encounter from abuse of LSA; High Blood Pressure and HPPD. High blood pressure is temporary and can be solved with baby aspirin, HPPD is a beast of mental nature that will never leave your eyes, nerves, and brain. It essentially permanently alters nerve pathways in a completely screwy way, adding permanent static into your field of view. Use at your own risk.

People suffering with depression have been shown to be positively affected by microdoses of LSA over a long period of time.

Hawaiian Woodrose Seeds – Argyreia nervosa.

Acquiring LSA Containing Matter

This is incredibly easy to do and rather cheap. A quick google search will bring you to any number of botany, gardening, or smart-shops. I recommend highly purchasing Heavenly Blue Morning Glory seeds, from an organic garden supply shop. Mail this to a P.O box, because from here things get sketchy.

Legality

Having seeds in your possession is a legal can of worms. Never EVER have this linked to you. Being caught with seeds is not a crime in and of itself, but the D.A could argue intent to extract which would land you a schedule III charge. Do not get caught. If you are caught with the extract they could call "precursor to LSD" and you'd catch a murder charge.

Experience

There's no way to explain LSA to someone who's inexperienced with psychedelics. It's like being at the top of the earth, ideas just flow like water, everything has this beautiful glow, if you do enough and close your eyes, fractal visions of ancient temples in outer space twist at speeds you never dreamed possible. Everything is different on it.

For people who have experience with LSD, it's much less visual, more euphoric but less 'power tool to the brain', and way, way easier to think on.

Those with THC experience may find that it's a lot lighter on the dome, so to speak, but vastly more visual.

I do not recommend it as the first psychedelic that you do.

Process

I am going to go through the process for extraction. This is a cookie cutter template that will work for any source of LSA, and any amount of LSA. LSA is extremely sensitive to light after grinding, do not expose to U.V radiation, and preferably do this in a Red Room. I will hereby refer to source matter as prelsa.

1. Grind prelsa in any electric coffee grinder. Do not grind too fast, any amount of heat will decay some of the LSA. Use pulses of grinding until it is a fine power.

Sleepy grass – Achnatherum robustum.

Ololiúqui – Turbina corymbosa.

2. Place very high purity (80% and above) grain alcohol into your freezer. Set freezer temperature as low as possible. It will be extremely cold. You may be tempted to drink it when you find it is not frozen, if you do this your esophagus may freeze and shatter and you will die. The colder the better but conventional freezer temperature should work fine.

3. Place ground prelsa matter into a coffee filter, and let it sit in a glass of the extremely cold alcohol. Let it soak for several hours in the cold. You may now remove and trash the prelsa matter within the filter.

4. Pour the solvent into a Pyrex Rectangular Storage Box. Evaporate in a ventilated but cool area. You may do this under a stove hood with dry ice sitting underneath the box to keep the LSA stable.

5. Scrape and Enjoy! You may mix with cold water and drink, mix with everclear and drip onto sugar cubes, or sublingually consume the residual LSA. The product left over should be 80-99% pure depending on source matter.
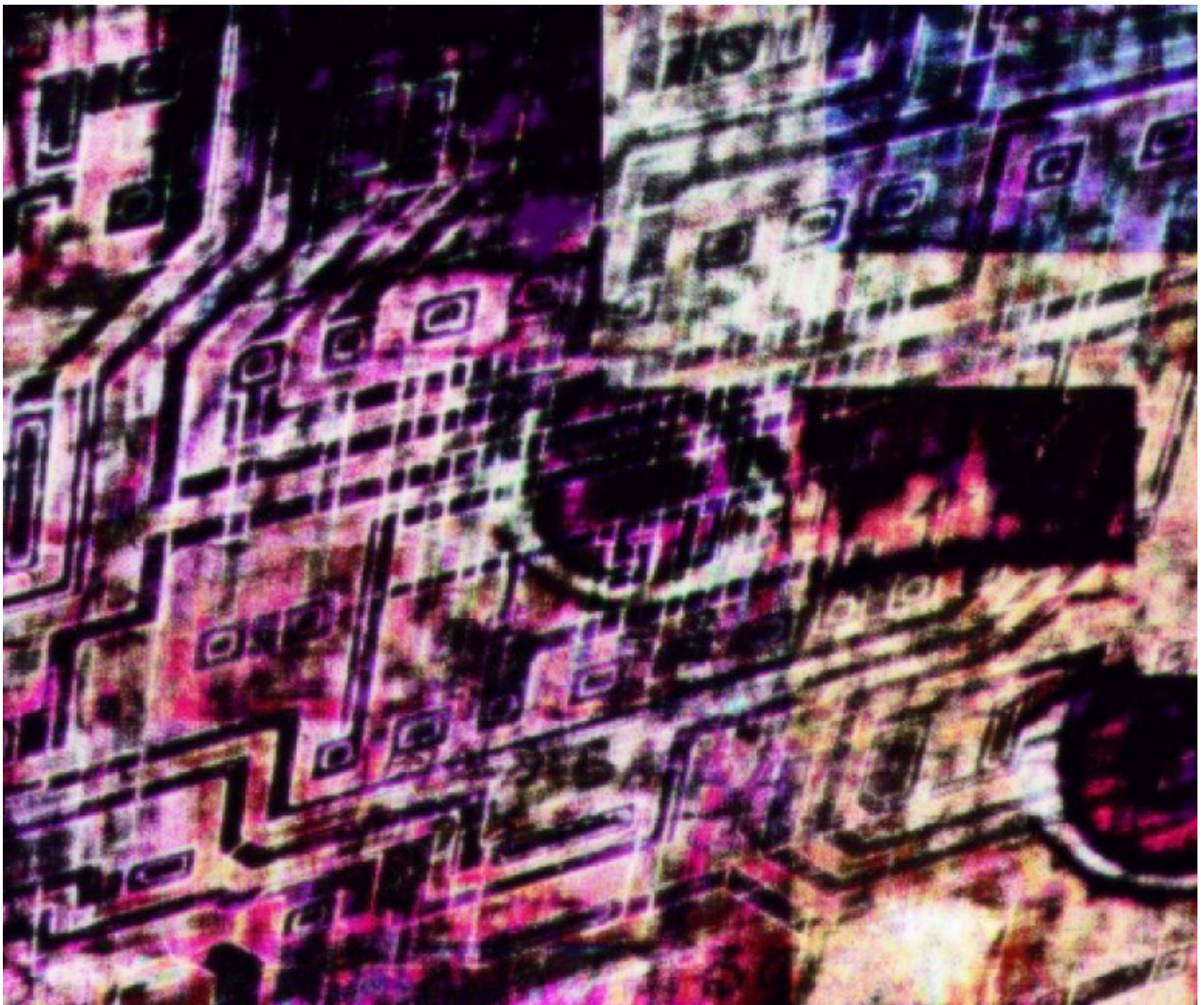
Written, experienced, and distributed by 'czech'

Pre LSA

Post LSA

Private

By Tom Millicent

I was receiving calls from a private number - I had answered months ago, the sound of a machine (perhaps a computer) could be heard; whirring and clicks - but there was no response.

A dream spirit took my soul hostage - it gave me three options: you leave now, I will destroy you; stay: I will devour your soul; give me your, soul I will contact you.

I woke as my phone was ringing - it was a private number, I couldn't wake up fast enough - again there was no message...

These Images scattered across this spread.
Each is a part of the storry...

...These are SSTV test transmissions.
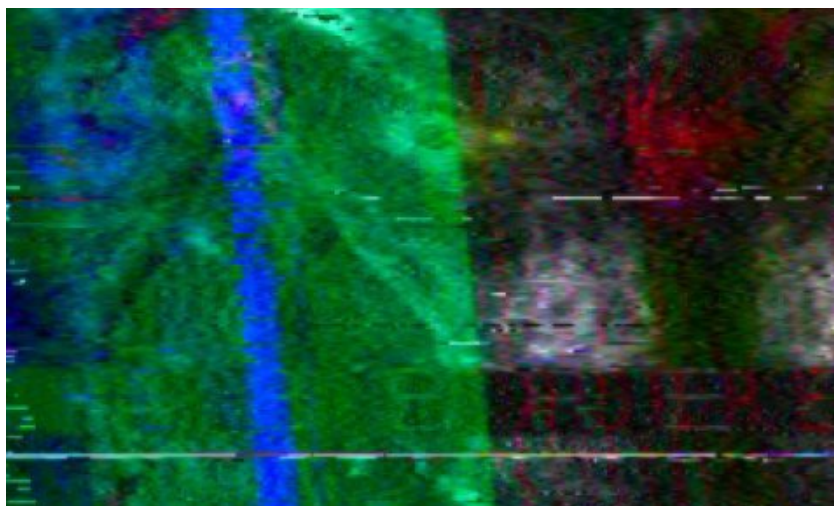Some with and some without autoSlant...

...These are send and recived using a Laser setup created to transmit audio...

...The intensity of a laserdiode gets amplitude modulated by the output of a PC soundcard...

...The beam hits a Photodiode on the next building connected to an other PC reciving and decoding the signal...

Recommended Reading

By FORMAT

"The Art of UNIX Programming" by Eric Raymond is available here: http://catb.org/~esr/writings/taoup/

"The UNIX-HATERS Handbook" by various writers is available here: http://web.mit.edu/~simsong/www/ugh.pdf
Also see this:http://www.art.net/~hopkins/Don/unix-haters/login.html

This recommendation is different from the others, as it recommends two works this time. The former, TAOUP, isn't being recommended for its merit, but, instead, as a work to be glanced over and reviewed while reading the latter, TUHH, which is the main subject this iteration.

The UNIX-HATERS mailing list is such a USENET list for rants concerning UNIX and its various deficiencies. TUHH can largely be considered a commentary on UNIX interspersed with many such rants. It is well-written, flows nicely, and can be read in a matter of hours, especially if the reader has already used UNIX.

This work is often dismissed as old and thus invalid, but perhaps the strongest point is how many of the complaints are still valid today. Arcane syntax and rituals, preventable failures, lack of fundamental features, an unforgiving nature, and the emulation of hardware from the 1960s (and more!) is still very much alive in the UNIX tradition as seen today. Much of this is contrasted with other systems of the time that avoided most or all of these issues.

The fourteen chapters detail:
• the birth of UNIX
• how it treats its newest users
• the documentation issues, when documentation exists
• mail frustrations
• the "joys" of using USENET
• the terminal and all of its quirks on top of quirks
• X-Windows or "This is not the ultimate window system, but I believe it is a good starting point for experimentation." [1]
• the "tools" UNIX provides for programming
• programming in the UNIX environment
• C++
• system administration as system babysitting
• a lack of security
• the unstable filesystems
• NFS

Looking at this list, some annoyances have largely disappeared, some have only been exacerbated, and others have been abstracted under other programs that usually work, but not always. A great deal of the fun is finding exactly what these are. To its credit, the GNU project has eliminated many implementation frustrations in UNIX life, but what wasn't caused by implementation was caused by design, which GNU has been less hostile towards.

> " *Two of the most famous products of Berkeley are LSD and Unix I don't think that is a coincidence* "

References

[1] http://www.talisman.org/x-debut.shtml
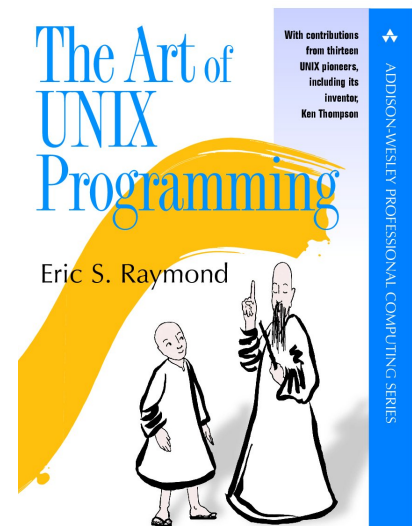
[2] Observe a concatenative language for an example of easily passing output of one procedure as input of another. The "pipe" is nothing innovative.}, being UNIX or simply heavy-handed implications of such. Many things considered good are said to have been

TAOUP is recommended reading as an example of the "UNIX weenie" archetype. The entire work should be read with a heavy skepticism, as disadvantages and ills of UNIX are understated, ignored, or played as positives while the advantages are grossly exaggerated. At times, there are lies concerning the origin of concepts, such as "Open Source Software" or the concept of the pipe. [2]

The two works share topics in parts and the difference in perspective is interesting, to say the least. I recommend reading all of what the two works have to say about sendmail. For good reason, TAOUP largely sidesteps discussing actually programming in UNIX.

The twentieth chapter of TAOUP does explain some issues with UNIX design, but often handwaves them away.

With regards to a UNIX file being an ordered collection of bytes:

"On the other hand, supporting file attributes raises awkward questions about which file operations should preserve them. It's clear that a copy of a named file to another name should copy the source file's attributes as well as its data - but suppose we cat(1) the file, redirecting the output of cat(1) to a new name?"

Vomiting a file is a very interesting operation, which usually requires **reset** be used to fix the terminal or **stty sane** when **reset** fails, which occurs often enough to require knowing this. The tradeoffs between supporting file metadata and how this affects vomiting files is an ongoing discussion to this very day.

If you find yourself skeptical of my recommendation, observe this [3]

The UNIX-HATERS Handbook
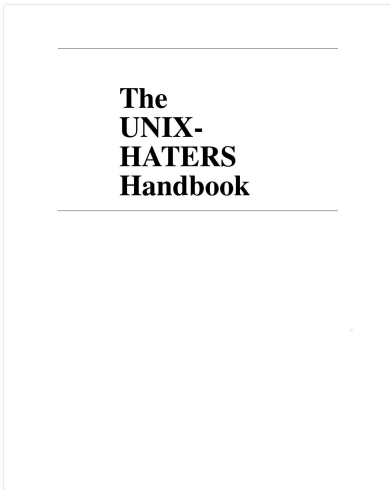
FCNTL(2)   Linux Programmer's Manual   FCNTL(2)

...   By default, both traditional (process-associated) and open file description record locks are advisory. Advisory locks are not enforced and are useful only between cooperating processes.
Both lock types can also be mandatory. Mandatory locks are enforced for all processes. If a process tries to perform an incom?

...
BUGS
...
  Mandatory locking
      The Linux implementation of mandatory locking is subject to race conditions which render it   unreliable: a write(2) call that overlaps with a lock may modify data after the mandatory lock is acquired; a read(2) call that overlaps with a lock may detect changes to data that were made only after a write lock was acquired. Similar races exist between mandatory locks and mmap(2). It is therefore inadvisable to rely on mandatory locking.}

"implicit" in the "UNIX tradition" and many things considered bad are said to have been part of the reason other systems have failed.

[3] {Issuing man fcntl on a GNU/Linux system should display roughly this man page, probably.}:

Set Theory
By popefucker

What is Infinity?

That's a really hard question to answer.

If you ask people on the street, they'll probably tell you it's a number. Many people use it that way; a lot of us remember telling our siblings "Oh yeah? Well I have infinity plus one!" Luckily, our ape-brains tell us, you'll never need to comprehend infinity; it's so vast that you'll never see all of it, no matter what you do.

Let's assume infinity is a number. Let's assume it is the *biggest* number, such that all numbers are less than infinity, except infinity itself. We can express that mathematically pretty easily (R here refers to the real numbers, which is basically the same as 'all numbers'):

statement 1:   if $n \in R$, then $n \leq \infty$

Let's say another obvious thing: when you add two numbers together, you get another number. This can be proved pretty easily by induction.
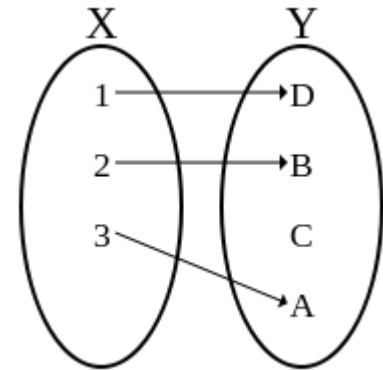
statement 2:   if $a, b \in R$, then $(a + b) \in R$

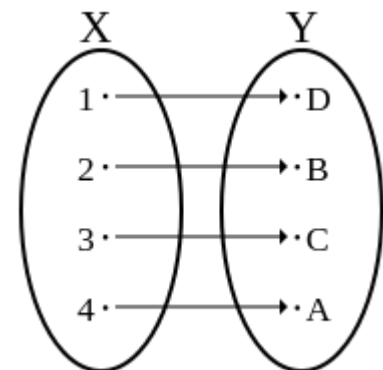And lastly, a blinding flash of the obvious. When you add a positive number to something it gets bigger.

Statement 3:   if $a, b \in R$ and $b > 0$ then $a + b > a$

If you paid a lot of attention in math class, this part will be obvious. Now we'll say: if all of those are true, then what's $\infty + 1$? We know it has to be real because of statement 2, so we know it has to be less than or equal to infinity because of statement 1. But, we also know it has to be greater than ∞ because of statement 3. Thus, we've found a contradiction. Infinity cannot be a number, at least not in any number system that follows the rules of the real numbers.

So what is infinity if it's not a number? To answer that, you have to start talking about sets.

Sets

Sets are pretty easy to understand, in their basic form. They're just groups of numbers. {1, 2, 3} is a set. And, as you can see, it as three members: 1, 2, and 3. Easy.

They get a little harder when you start talking about infinite sets. Let's talk about the set of natural numbers, N. A natural number is one of two things: 1, or the sum of 1 and another natural number. It's clear that no matter how many natural numbers you have, you can always make another one by adding 1. So, we say that the set of natural numbers N is infinite – that is, there is an infinite amount of things in the set. We call the amount of things in a set its cardinality, and it's important to note that the *cardinality* of a set doesn't need to be a number, only the set's elements themselves need to be.

This is an injection

This is a bijection

This is not an injection

By definition, the set of natural numbers has a cardinality of Aleph-null, which we use as a baseline. However, there are bigger infinities than Aleph-null, which blows our entire concept of a single ∞ right out of the water. How can one infinity(or infinite cardinality, more accurately) be 'bigger' than another?

The way we compare infinite cardinalities is by looking for something called an injection. An *injection* (or injective function) is some sort of way of mapping every member of one set onto exactly one member of another set. If you can find an injection from an infinite set to the set of natural numbers, then we say it is countable (don't read into that word, it is just a convention). A bijection is a special kind of injection that is an exact one-to-one correspondance between two sets. If you can find a bijection between a set and the natural numbers, then we say that that set also has a cardinality of Aleph-null. It is very likely (but not known) that Aleph-null is the smallest possible infinite cardinal.

A surprisingly large amount of known sets have cardinality Aleph-null, and many of them have very tricky bijections (such as the proof that the rationals are countable). However, one infinite set that does not have a cardinality of Aleph-null is the set of all real numbers R, which we saw when proving that infinity is not a number.

The Diagonal Argument

R (or "the continuum") includes every number that does not have an imaginary component. It includes every number that can be defined, and many more that cannot. The proof that R has cardinality greater than Aleph-null is very simple. Suppose you took an infinite set of real numbers, and mapped them to the natural numbers, like so:

1.002029189877857...	$\rightarrow$ 1
2.928984884893992...	$\rightarrow$ 2
99349.02939191911...	$\rightarrow$ 3
...	

The actual numbers don't matter, but they have to be non-terminating. However, now that you have your infinite list, you can construct a new non-terminating number by taking the n'th digit from the n'th element in the list, and making the n'th digit of the new number different. Thus, the new number will always have at least one different digit to every number already in the list, and it cannot be in the list. Since the list was already mapped perfectly to the natural numbers, the set of all real numbers (which includes the new number) must be larger than the set of all natural numbers, and thus must have a cardinality greather than Aleph-null. So, some infinities are greater than others, and there are an infinite amount that are greater still.

So, did we really figure out what infinity is? Well, no, not really, but we found out it's even more confusing than we thought. I guess sometimes that's the only victory you can claim.

Sidestepping Wifi Trials

By Vixn

Introduction

You're at the airport and need some Wi-Fi. Luckily for you the airport provides you with some, but only for an X amount of time. When that time runs out, you're off the net.

You're scanning for local Wi-Fi networks in your neighborhood and you happen to live in a city where a company such as COmc4st has happily commandeered your favorite neighbor's router to provide a hotspot that you can use for a not-so-small fee. Luckily for you, after poking around, you notice that they offer a one-hour free trial for their services. After that hour, they kick you off.

Here's a simple guide to sidestep that time limit. In essence this tutorial can be given in one sentence, but I've opted to write it in such a way that I hope is informative and encourages a certain kind of critical thinking so that a newcomer might progress and come up with tricks like this independently.*

The Trick

When you try reconnecting to a hotspot like the ones mentioned above, the router of the hotspot provider rightfully identifies that your computer has already taken advantage of their free trial. The first question that we might ask is how such a hotspot can identify your machine as the same one—even after a reboot, etc. The answer (generally) is that your machine has thing called a MAC (Media Access Control) address. This has nothing to do with Apple. Basically, a MAC address is associated with your network card and allows a router to identify you. It normally stays the same and consists of six groups of two hexadecimal digits usually separated by colons (e.g., aa:bb:cc:dd:ee:ff). This address can be changed (read: spoofed) pretty easily depending on your system. When you spoof it, you temporarily give your machine a new address thereby telling the hotspot that your computer is someone else. And really, that's all there is to it. Spoof your MAC address.

How to do it exactly depends on your operating system, and tutorials are easy to come by. One thing to note is that not all MAC addresses work if randomized improperly so don't be discouraged if your first attempt fails. The reason it doesn't work is likely because the second digit of the first grouping of your spoofed address (the second 'a' in the series above) must contain only a 0, 2, 4, 6 or a, c, e.

* I in no way to claim to be 1337. And I understand that many of you likely know this little trick, but I thought I'd share anyway. This is info that I wish I knew earlier as it has saved me quite a bit of cash over the years.

Further Tricks

The next steps you might consider are writing some scripts. One idea you might want to try out is having your system randomize your MAC so that the second digit in the first grouping takes the digit limitation into account. Or you can randomize everything else, setting that second digit permanently to one of the digits mentioned above— but if you take advantage of hotspots daily it might leave a calling card you might not want. Another idea would be to write a script that turns off your network card, spoofs your MAC, and reconnects your machine for you. This is a good idea because this process needs to be done anyway each time you spoof your MAC and need to reconnect to the hotspot. You may even want to add more on to the script to aid this process. For instance, you might want it to auto-fill and randomize user info should that might be requested during a free trial sign up.

The downside to all this is having to reconnect each time the trial timer runs out, but it's better than paying. Writing some code speeds things up.

Uses and Abuses

Obviously this trick can be taken advantage of in various ways. Before going on let me just remind you that your data is flying through the open air to some random person's hotspot and you should take some steps to protect yourself (e.g., utilizing a live OS, TOR, a VPN, HTTPS, etc).

Without going into too much detail these hotspots can be used creatively to your advantage in some of the following ways:

- Save money by not paying for Internet (if you live in an C0mc4st xf1n1tyw1f1 infested city, for example).
- Save time by not running software to steal a Wi-Fi password.
- Further anonymity by having no active Internet services in your name.
- Further privacy by avoiding high-density public spaces and cameras (rather than utilizing, say, a coffee shop for certain activities, find a quiet area with a hotspot) (note: high traffic areas might also be used to your advantage).
- Fast and easy access to practically an unlimited source of regions in your city for your Internet doings.

Hope this helps some of you and fills you with warm digitized feelings of being a cyberpunk. If you ever want to talk you can find me on #lainchan or Tox:
(4C373BB3FCB672F24E0E58B9F8A40C7EA6630F0166F37A56D9 5A1D133FB44C3C8D48304EA4AD)

Trip Report
By Anonymous

- **Substance: 4-aco-dmt**

- **Dosage: 20mg (approx)**

- **Route of administration: Oral**

This trip report is a description of my first experience with 4-Acetoxy-DMT. This was my first experience with psychedelics in general. A couple months before the events described here I had becomes very interested in psychedelics. I had almost no experience with drugs of any sort other than alcohol and opiates, so psychedelics struck me as being very exotic and interesting. After visiting several forums as well as Erowid, I learned about research chemicals. I was told that 4-aco-dmt was an enjoyable substance; so I went ahead and purchased a small amount. After this I continued reading advice/experiences on the internet and started to make some preparations.

I collected music that I liked and arranged it into a playlist that I could listen to during the comeup and the trip itself. I also went around collecting some "trippy" images that I thought I might enjoy looking at and some videos that I thought would be cool (Specifically the TV show "Off the Air.") To be honest, I was very nervous about my first trip. I decided that I would go for a walk along the beach near my house during the trip. I didn't have anyone who could trip sit for me, and I wasn't entirely sure exactly how lucid I would be while I was tripping. Still, I was set on doing it anyway. In retrospect this wasn't exactly brilliant. Walking around in public while tripping is fun, but very risky. Live and learn I suppose.

- **Now for the trip itself.**

It was a Friday night, and the weather was very pleasant (I had settled on night time because I was worried about going outside during the day while tripping.) I sat down at my computer desk and took out the 100mg baggie I had acquired. At the time I did not have a scale capable of measuring milligrams, so I did my best to divide the powder into five even lines for a 20mg dosage (This was another rookie mistake. I would not suggest eyeballing dosages.) I scraped the drug into a small and thin slice of cheese, balled it up, and then swallowed it whole as if it were a pill. I was extremely nervous, my hands were shaking and my heart was racing. I turned on the playlist I had arranged and sat back breathing deeply to calm myself down.

My stomach was empty, so I started to really feel effects after about twenty minutes. I was hit by a wave of euphoria and began laughing for no particular reason. All the anxiety I had previously felt was gone, and I just felt wonderful. I looked around my room and saw that the white door to my closet was now tinted a greenish color (the same color as the blanket on my bed), which I found very funny. After some time I began to notice slight tracers from my hands moving, and decided it was time to
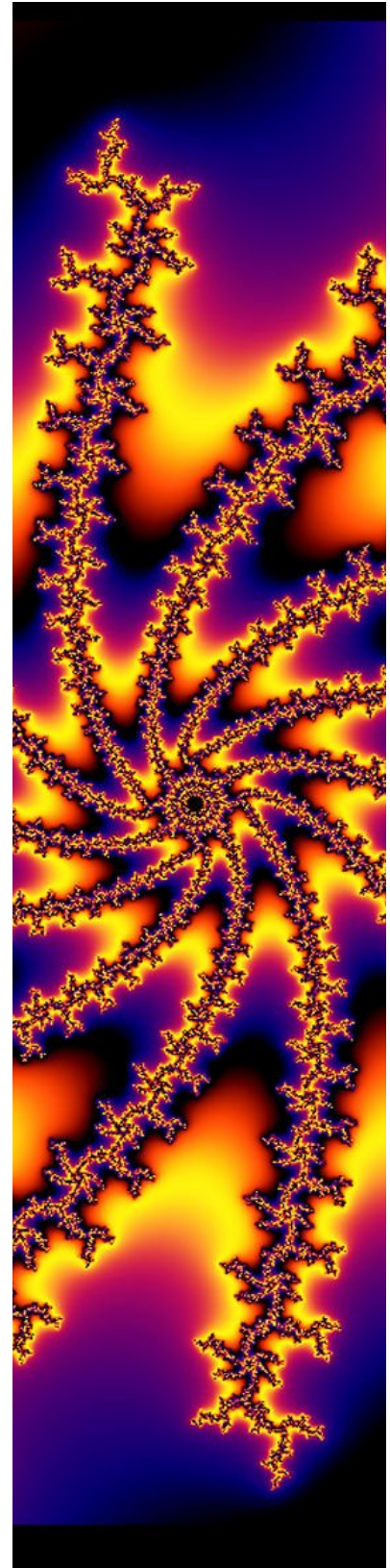
go out. I got up, went to the door, and opened it, which made me feel somewhat proud of myself for properly operating a doorknob. I walked down the stairs and stopped to pat my dog, all of which made me feel very good about everything in general. I was feeling very happy and content, like all was right with the world.

I opened the front door and immediately noticed that everything was very bright, despite the fact that it was around 9:30pm. Street lights seemed much brighter than normal, and the glow of the lights over the city were also very bright. The sky was beautiful. I went to the beach and stood on the sand marveling at the ocean and the sky. It seemed like every time I looked back at the sky the clouds and the glow from the city lights had changed color. The stars in the sky seemed to be moving around and I was amazed at how…large the sky as a whole was. I was laughing uncontrollably, and I was aware that there was a young couple sitting on the beach wall about 30 yards away. I didn't really care at the time, even though I felt like they were watching me (they probably were). I heard the girl laugh, which made me laugh even harder. At the time I believed she was laughing for a very long time, and the sound was very loud, as if she was standing right next to me. The sound seemed to echo inside of my head for some time. I estimate that I stayed where I was for about 45 minutes. My perception of time was pretty warped, and it felt like a much shorter period of time.

I suddenly decided that I wanted to run. There's a spot on the beach where the beach and the street diverge so street level is much higher up, so the beach wall is about ten feet tall.

I would say that I was probably about 150 yards away from there at the time. I started running as fast as I could towards that area, and I felt amazing. I felt like I was moving at an incredibly high speed. The beach seemed to stretch on forever in front of me, and I started to wonder how long it was going to take me to reach the end. Suddenly I looked to my left and saw a man walking along the street by the concrete beach wall. This startled me for some reason, and I realized that I was acting very strangely. I slowed down and tried to pretend that I was just jogging or something. At this point I was right at the area I had been trying to reach, so I went to the base of the high wall. While I was walking over to some of the large rocks at the base of the wall, I spotted a piece of driftwood. I said out loud "Is that a cat?" and went over to investigate it. I had previously been thinking about animals (specifically my dog) and I had wanted to touch a cat as well. I was interested in tactile sensations in general. Despite sprinting 150~ yards I didn't really feel worn out, but I was breathing heavily, which is interesting to note because I don't exercise frequently.

I went over to some large rocks that are right up against the base of the wall and began touching the concrete to see what it felt like. I really enjoyed the rough feeling of it, so I spent several minutes just running my hands along the wall. Eventually I stopped and decided to sit down. I sat there staring at the ground for a while and touching the rock I was sitting on. For some reason I got a slightly paranoid sense that someone was watching me, so I kept twisting around to look at the top of the wall to
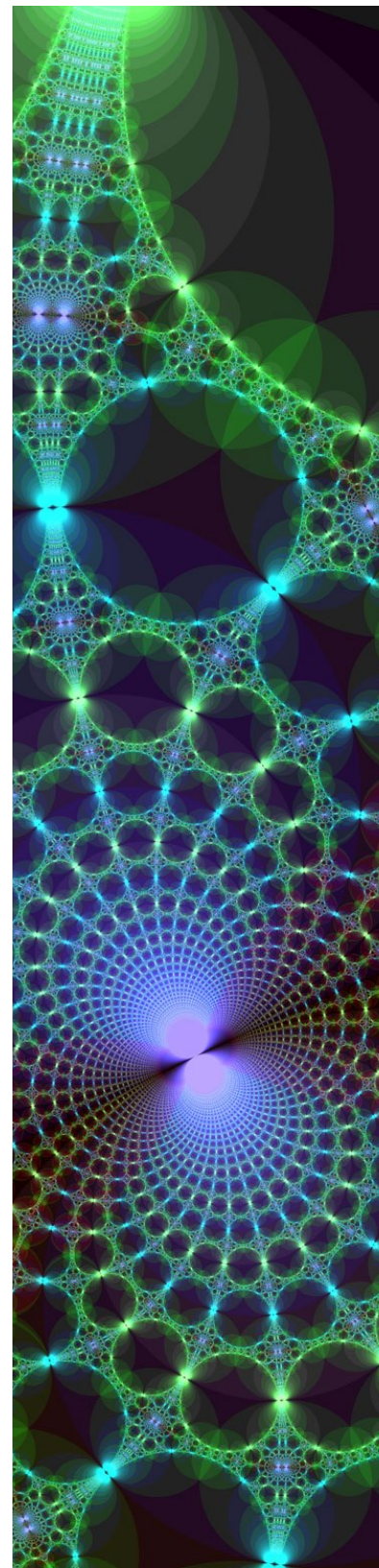
see if someone was looking down at me from the street. I sat there for a while admiring the ocean and sky some more. I tried to close my eyes to rest for a bit, but my memory of the scenery around me was so vivid that it was almost like I was still seeing everything even with my eyes shut. I reached up to touch my face and verify that my eyes were really closed, and they were.

I sat on the rocks thinking about my life and the people I knew for some time (I was also talking to myself the whole time, but that's not unusual for me). I reflected on my personal relationships, and felt that I knew a lot of very good people, and that maybe I didn't appreciate them as much as I should. My thoughts moved to a coworker who had recently quit. Despite the fact that I didn't know him very well, thinking about this made me very upset. I almost cried when I realized I would probably never see them again. My thoughts took a more depressing turn from there, and I realized I needed to steer myself away from that sort of thing.

After steering my mind away, I started to think about the people online who had given me advice. I started thinking about how I needed to thank them for everything they told me. I got up and started walking to an area where the wall gets a bit lower again, and looked over it while standing on my toes. I believe I saw a black and white cat standing on the wall, but now that I think about it I wonder if it's possible that I merley imagined it. I was very excited to see it, because I wanted to touch it and say hello. I tried calling out to it, but it just stayed where it was and looked at me. I then attempted to climb over a lower (5ft~) part of the wall, which should have been a pretty easy task. My coordination seemed to go out of the window though, because I ended up just sort of bumping into the wall instead of hoisting myself onto it. The cat then began to walk away, and I decided to go around the high area and back onto the street. Despite this, I suddenly noticed a clam shell on the ground and picked it up. I stood there examining it for a moment before I remembered what I had been doing.

By the time I got to where I had seen the cat, it was long gone (if it was ever actually there in the first place). I stayed standing on the sidewalk leaning against the wall, and I began to admire some trees on the other side on the street in someone's yard. Their beauty was breathtaking, and while I was standing there staring at the trees they seemed to fill my entire vision, almost as if my eyes were a camera zooming in on them. I was seeing an illusion of faces in the leaves of the trees, and I remembered seeing a painting showing something similar. I was amazed by this and kept repeating to myself "It's just like the painting!"

While I was standing there I noticed that someone was approaching me on the left. I believe it was a man wearing a hooded sweatshirt, but
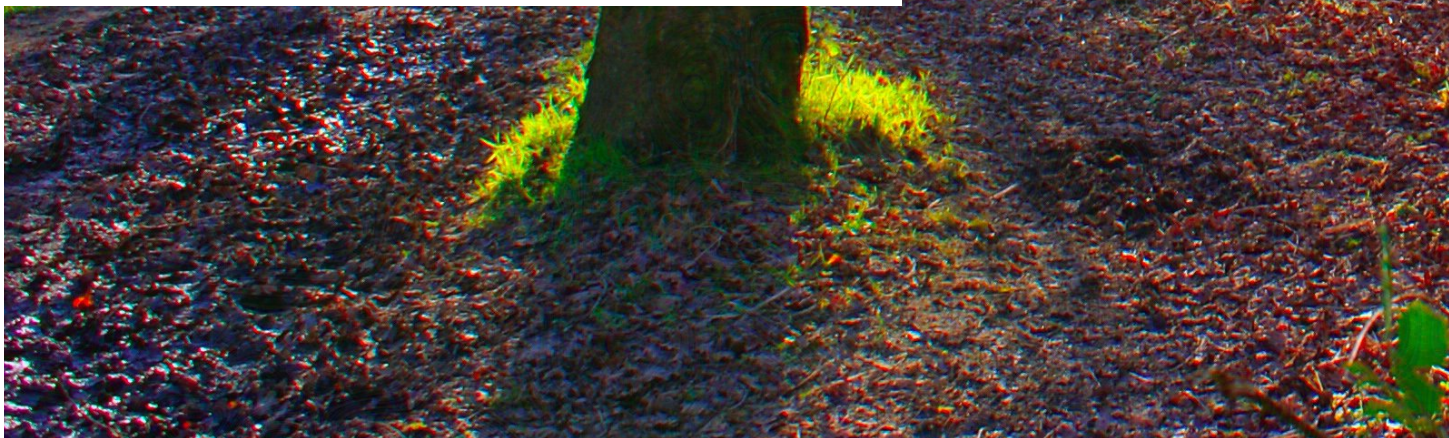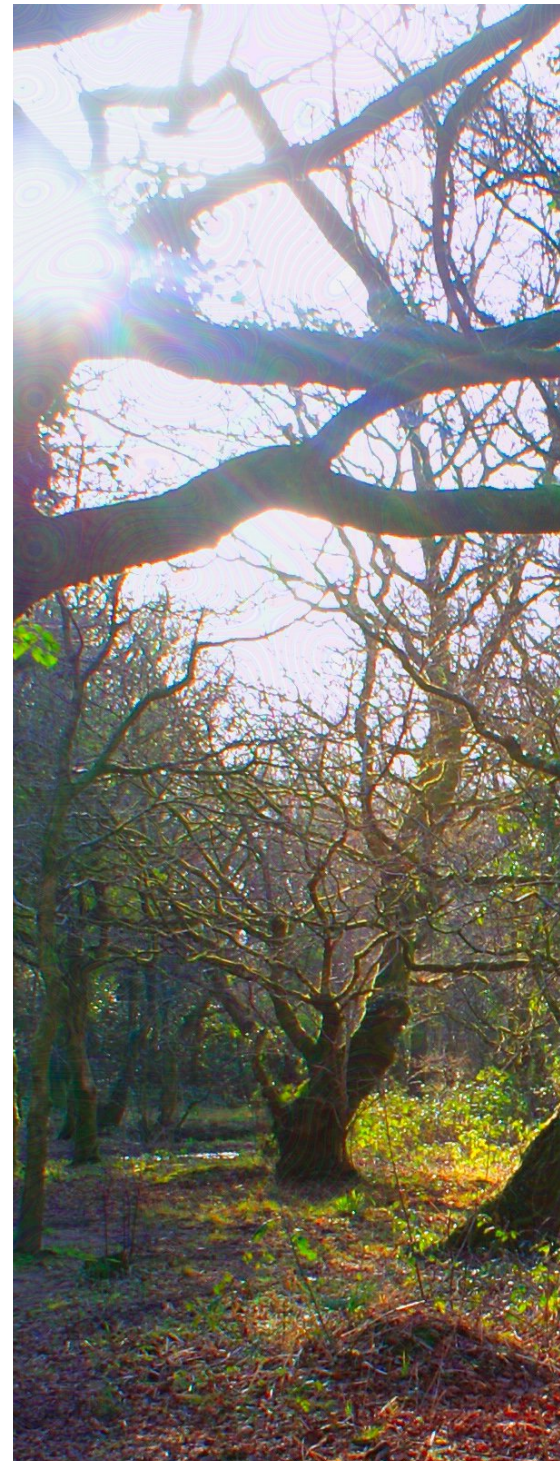
when I looked up, he suddenly turned around and began walking in the opposite direction very quickly. I had the sudden urge to go talk to him and at the same time I wondered "Where is he going?" and I was concerned that I had somehow scared him. At this point I realized that I had been standing on the sidewalk for quite some time with my mouth hanging open (I had actually drooled on myself a bit.) I then started to get paranoid about being arrested and decided I should walk home. When the idea of being arrested/confronted by the police crossed my mind, I started to get very annoyed at the notion of someone trying to arrest me for harmless fun. I quickly put this out of my mind because it was bothering me.

I started walking along the sidewalk to head home. I was making a conscious effort to act normal, because I felt like people were watching me. Still, while I was walking I noticed tracers out of the corner of my vision from my hands swinging at my sides. I began waving my hands around in front of my face to get a better look at them while walking. I realized that looked very odd, so I stopped. I continued to touch bushes, other plants, and grass on my way back though, just to see how they felt. Everything (especially plants and grass) had this wonderful color enhancement. Colors seemed supersaturated and had a sort of fuzzy colorful glow to them. This glow was not necessarily the same color as the thing I was looking at.

Right as I was arriving home, I saw my neighbor stepping out of his car. This freaked me out, because I didn't want him to know that I was on drugs. I started thinking to myself about what to say and how to act. This was pretty successful as far as I can tell, because I passed him and he said "Oh hey, what're you up to?" and I just said that I was out for a walk and told him to have a nice night. He said bye, and then kept walking. I felt very proud of myself for not panicking and immediately went inside.

After this, I went back to my room and looked at some more pictures, listened to music, and watched some of the shows/videos I had prepared earlier. All of which were very interesting and enjoyable. In total I would say the real trip only lasted about 3-3.5 hours, but I was still feeling a mild headspace for some time after that. It's hard to describe, but it was a sort of lightheaded and "funny" feeling in my skull.

Overall, the trip was amazing. Although I did feel paranoid or upset at times I would not say that these portions of the trip subtracted from the overall experience in any significant way.

I felt great for the next couple of weeks. I had plenty of energy and I was in a very good mood. I often found myself laughing at the absurdity of being, or just admiring the clouds in the sky.

Upgrading Physical Security Without Spending a Cent

By Nils Iwakura

Physical Security

You may have good security on the Wired, but that wont help you when someone breaks into your base of operations and steals/breaks your setup.

This is how to modify your locks to better resist lockpicking, without having to replace your keys.

Tools you will need are:

- A dremel/high speed rotary tool with collet for the diameter of the pins (Fig.1)
- Something to hold the dremel in place (I used a vice, you can clamp it to a table, just get it stable)
- A pin tray (I made mine out of cardstock that I accordian folded to give me ridges in which to distinctify which set the pins were from (Fig.2))
- Screwdriver or other tool to take out retainer. (Fig.3)
- Tweezers (Fig.3)
- Mini Files. I only used the edge of my thinnest flat file, and the corners of the triangle file. (Fig.3)

Your lock MUST have a removable core. Most doorlocks and high end padlocks should have this. To get to the core, you must take the lock out and get to the back of the cylinder. Different locks have different retaining methods, but the example I used has two screws in the back that you must remove (Fig.4) (another common method is a C-retainer). Remove the retainer, insert your key and turn. It should be able to come out the front, but be CAREFUL, the driver pins will be under spring tension and will jump out and you may lose them, or lose where it was (my example has uniform driver pins, some dont).

At this point you should make/have a pin tray. If you mess up the order, the original key will not work and youll have to try to guess which pins go where based on the key.

Hold or vice up the lock so that the springs and driver pins are coming out upward, turning the key so the key pins are also trying to exit upward (for mine I just turned the lock upside down, and turned the key to be the right way up). Use your finger to cover the back of the hole as you slowly pull out the key with the cylinder. Once you hear the click of the driver pin jumping out of its place, take it out and put it in the tray (this pin should be the very back pin, so it goes in the highest pin number.

Fig 1) A dremel/high speed rotary tool.

Fig 2) A pin tray.

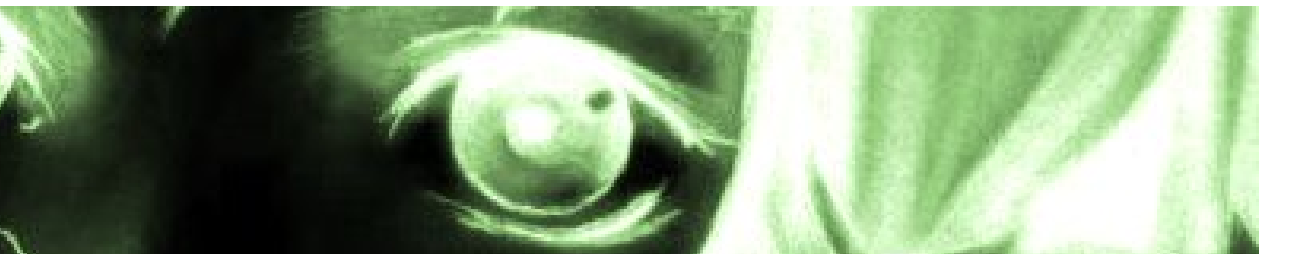**Fig 3)** Screwdriver or other tool to take out retainer

Fig 4) Remove the retainer

Fig 5) Put a pin in the dremel, trying to leave as much exposed as you can.

Fig 6) Aim to cut serrations or spool shapes into the pin.

Mine is 5 pin so I would put it in the 5th position in the pin tray). Repeat, putting pins in their proper place in the pin tray until theres no pins left (Fig.2). Fully remove the keyway, careful to keep it upright as to not lose the key pins. Carefully remove the key and put the key pins in their respective spot in the pin tray (Fig.2). You may optionally flip the lock body upside down to dump all the springs out, but if its in a vice you dont need to worry about it, as long as you dont tip it over and end up losing a spring. If your pins already have spool, serration, or other security features, then you/your landlord/previous tennants chose a decent lock. You can still cut whatever standard pins are in that lock, but leave the already cut ones alone.

Now comes the fun part. Take a pin and tighten it up in the dremel, trying to leave as much exposed as you can while still having a good hold (if the pin in a key pin, put the side with the 45 degree end into the collet, and cut the other end) (Fig.5). Set the speed to low if you can, turn it on and let it rev up. Now you start cutting into the pin with the files, careful not to touch the ends of the pin (if you do, youd need to rekey). Aim to cut serrations or spool shapes into the pin (Fig.6). This will make it harder to pick by giving false sets. Repeat this process with as many pins as you like (I did it with all of them to get some practice in).

Once you've finished cutting your new security pins, you can start putting everything back together. Start by putting the springs back into the lock body if you took them out. Put all the key pins back into their correct places in the keyway, making sure the right side is down (45 degree end goes in first). Then take your tweezers, putting the driver pin in place before pushing it down and sliding the keyway over it to stop it springing out again (make sure you don't dump your pins by turning it upside down). Its best to put the driver pins in non-cut side first, as they'll false set more often. Continue putting pins in and sliding the keyway until all pins are in. Push the keyway all the way in before turning to lock it. Put the retainer back and test it with the key. If it opens, you're done! Put it back where it belongs and enjoy your more secure lock (or take it with you to DEFCON and try to frustrate the lockpick village).

If you've got questions or just think I'm a cool guy and you wanna talk to me, send something over to nils@tfwno.gf

I am not responsible if you go fuck up your lock or if you get hurt doing this. Some of my lock terminology might be incorrect, so forgive me, I'm not a professional.

Web scrapers, a guide and a horror story

By raxmur

A classic website can be thought of as a program that takes a set of information from its server, assembles it as needed and sends it back the browser. A good API (Application Programming Interface) is one of the most useful features a website could possibly have, because it exposes the raw information instead of wrapping it in its own UI, so that you get to decide how to assemble it and what to make of it.

Some websites unfortunately lack this great feature, perhaps because of developer laziness, incompetence or simply lack of interest. This is where the scraper comes in.

A scraper is like an API that you have to write yourself: it disassembles the webpages, extrapolates the required information and collects it into a data structure that you can work with. Scrapers might not be as powerful or as clean as normal APIs, but they're often the only way to get access to information programmatically.

Tools of the trade

To write a scraper, you're gonna need a few things:

- Knowledge of a programming language. I choose **Ruby** because it's concise, has a REPL, and great libraries, but you can choose any language you see fit.
- A web browser with developer tools. All major browsers have these, so just pick your poison.
- curl. This is an invaluable tool for whoever works with the internet.
- A library to make http requests. I use Net:HTTP because, well, it ships with Ruby.
- A good library to access XML, possibly through XPath. This is gonna be the deal breaker for the language you're gonna use, because XML is a mess and you *will* have to deal with malformed XML. **Oga** is what I used. (Some use regular expressions to parse XML. I do not recommend it.)
- Some Javascript knowledge. Single-page websites often use their own internal REST API to fetch the information, so you may have to read through some frontend code.

A simple example

Writing a scraper is made up of two phases: first, you'll need to understand how the website works and where to poke to obtain the information you want, then, you'll have to parse the resulting webpage and select the interesting bits.

Let's take lainchan as an example: let's ignore that it has a perfectly good JSON API and say you want to write something that consumes the posts in a thread.

First of all, pay attention to the URLs. Websites are programs after all, so they (should) work in a logic way: try to find patterns by looking at the URL and the function of the page you're on.

Let's take lainchan, for example. In lainchan's case, this is very simple: the first element in the path is the board ID (Î», cyb, lit or what have you). Then, in the index pages, the second is the page number, while in a thread it's just a prefix, and the third is the ID of the thread.

You can write some functions to generate the correct URLs, if you want:

```
```
BASE = 'https://lainchan.org/'
def index(board, page)
  "#{BASE}#{board}/#{page == 1 ? "index" : page}.html"
end
def thread(board, id)
  "#{BASE}#{board}/res/#{id}.html"
end
```
```

Now that we got the logic down, we can begin the second phase. Locate the elements you want to scrape and right click on them. There should be an option called "Inspect Element", or something similar. This will open the Web Inspector and select the line in the HTML with the element you selected. Try to find a pattern in the elements' classes or structure.

Let's take lainchan again. Let's say we only want to get the opening post bodies for each index page. All OPs have class `op` and the body is inside a div with class `body`. Let's translate that into code.

```
```
%w[uri net/http oga].each { |l| require l }
doc = Oga.parse_xml(Net::HTTP.get(URI(index 'cyb', 1)))
ops = doc.css('.op')
op_bodies = ops.map { |post| post.css('.body').text }
```
```

With just a few lines of code, we were able to easily get access to the information we needed. Wasn't that easy?

A tale of bad code

Unfortunately for you, not all websites are as easy to scrape as lainchan. Sometimes you come across a huge tangled mess that might take you hours to unravel. Let me tell you a story about how not to write a single-page website.

dojin.co is a website that gathers download links to doujin music: Japanese indie music often inspired by videogames such as the Touhou series or Kancolle. It is amazingly heavy, for some reason; it is not rare for my browser tab to crash while I'm trying to search for something. One day I got tired of its bullshit and decided to write a scraper for it, and try to understand what was the cause of its abysmal performance.

Little did I know of the horrors lurking underneath.

My first guess was that the developer just did what everybody loves doing these days: throw a bunch of crudely combined JavaScript frameworks at it and call it a day. That's understandable: the site appears to have only one developer, and he seems to enjoy web design more than programming. But as I open up the dev tools, I am greeted by a relatively short list of unorderly files with surprisingly little trace of framework-like drivel. Guess the code really is that bad.

The codebase is kind of a mess, but it doesn't take me long to find the function called when scrolling down to load more albums. It makes a POST request to an `ajaxurl` with some settings and an `arraySet`, which seems to indicate what to send.

`ajaxurl` turns out to be `/wp-admin/admin-ajax.php`. This tells me that it's using wordpress underneath, and that the server side of the website is likely to be a huge mess of PHP. I open up a terminal and try curling the address:

```
$ curl -X POST http://dojin.co/wp-admin/admin-ajax.php
0
```

There doesn't seem to be anything stopping me from doing it. Does this mean that the authentication is useless? Most likely. Let's try curling with the parameters from `displayMore()`:

```
$ curl -X POST http://dojin.co/wp-admin/admin-ajax.php?action=infiniteScrollingAction&postPerPage=35&offset=0&arrayS
et=%5B34959%2C22539%2C16137%2C23916%2C34978%2C34982%2C3
4517%2C37926%2C1797%2C2453%2C2586%2C1462%2C11373%2C23913
%2C34984%2C5716%2C6372%2C7708%2C23910%2C2188%2C3441%2C1
7564%2C34980%2C37277%2C9111%5D
```

This returns a JSON file with the results in plain (malformed) HTML under the "data" key. Why you would wrap that in a JSON object is beyond my comprehension, but hey, it works for him. After playing a bit with these parameters I figure out that the numbers in "arraySet" are just album IDs. This is also a weird decision, but it sort of makes sense.

Later on, I find what seems to be the POST request invoked when searching: the payload is a hashmap containing some options and the search terms; the "artist" and "style" fields are represented by IDs. curling with those parameters returns the search results and an array with the album IDs. Bingo! Now all that's left is figuring out where it's getting the artist and genre IDs from.

I search the whole codebase for other AJAX requests, but nothing turns up. I try to run the profiler while searching and, while it does register an incredibly high call stack and a huge performance hit at various points, it surprisingly shows no signs of hitting the network. Baffled, I try using "Inspect Element" on one of the search suggestions, which is when the realization hits me like a bus.

THEY WERE THERE ALL ALONG! All the artist and style IDs where all hidden in plain sight in the page HTML.

It takes me a while to recover from such a sight. I start writing the scraper, but the front page is so huge that nokogiri, the XML library I was using before, chokes on it, and I'm forced to switch to Oga. After fighting with XPath and encodings, I finally get a JSON file with all the artists and their respective IDs. It is 84KB; given the verbosity of XML I estimate the source to be at least 700KB, and I'm not far from the truth: the whole front page weighs a little short of 1MB.

But why exactly is it so slow? Well, first of all, 1MB of HTML is a lot to digest for a borderline toaster such as mine.

Second, this 1MB document is being queried all the time. How do you suppose it shows the suggestions? Obviously it queries the whole thing for the elements with a name attribute that starts with the search term and changes their properties to make it so they appear in the suggestion list. Querying such a huge document takes time, especially since it has to run synchronously and thus hangs the page until it's done. Now imagine this running each time you push a button on your keyboard.

Despite this madness, the scraper turned out to be pretty simple to write: I made it generate a nice HTML file with the results. It looks a lot like the website itself, but without all the crap.

And this, my friends, is the reason why a good API is the best feature of a website: it provides a standard interface that gives you the freedom to access its information the way you want to, unconstrained by the idiosyncrasies and failings of its UI.