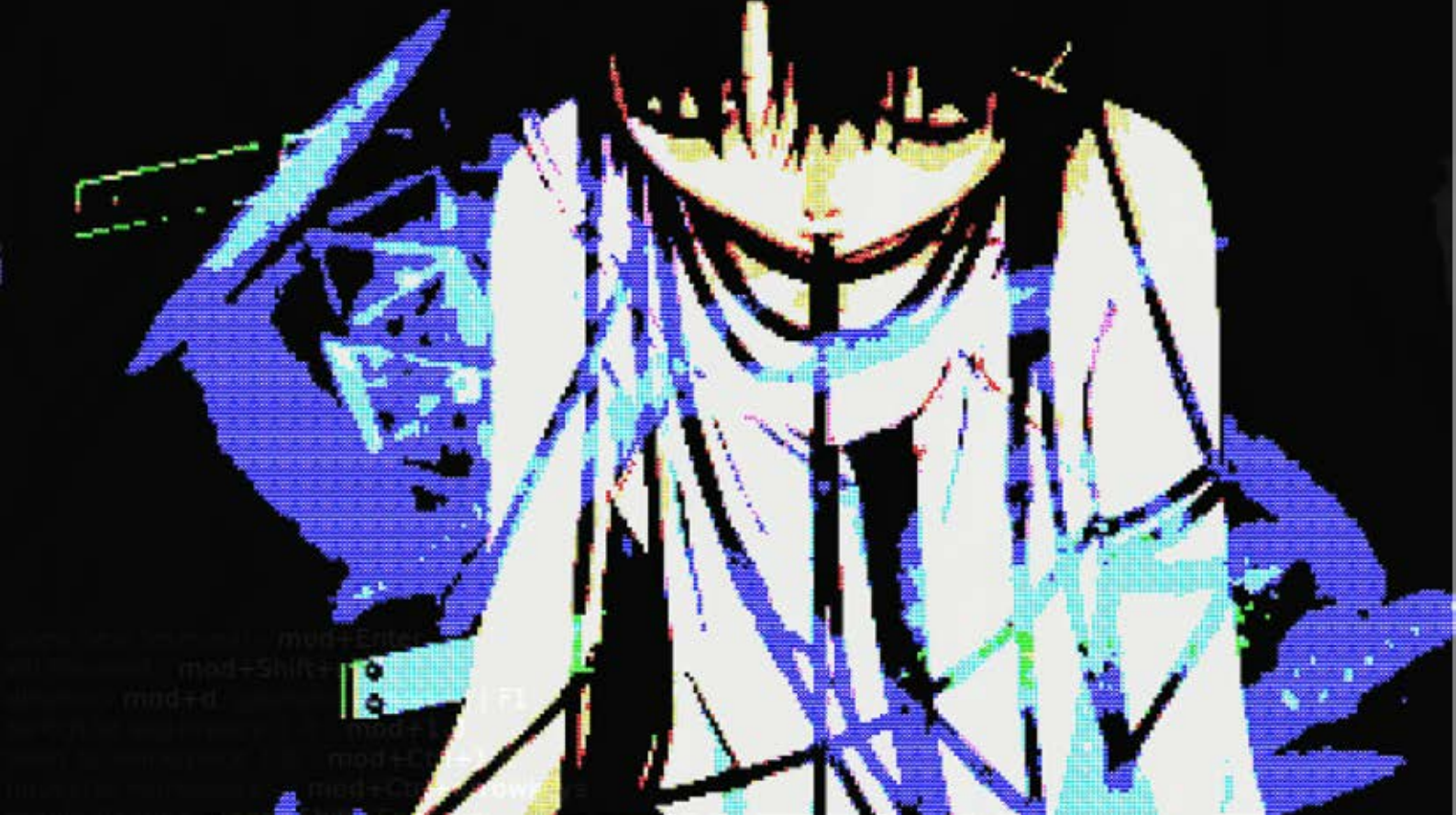
lain zine

volume 2



I read the last Lainzine and now I'm on the run from seven different governments, two of which aren't human. Truly this knowledge was not meant for mortal men. Looking forward to the next issue.

— Anonymous

Lainzine
Issue 2
Published 28 August 2015

Contents

COLOPHON
Created by the good people of Lainchan from all around the world.
https://lainchan.org

STAFF

Editors	Junk, kk7, A731, and Not Jesus
Artist	kk7
Typesetter	Ivan
Special thanks	Hash_Value, Kalyx, jove, darkengine, and you

Editor's Note

My apologies for the delay on this one. To ensure that the Lainzine is never late again, we will automatically accept any submissions that contain the phrase “quality over quantity”.

http://pastebin.com/PLqCfqM3

— Junk

Word Search

There are an undisclosed number of NSA blacklist words scattered throughout this zine. Can you find them all?

The Way of Schway

by serene

> *If the Wired were a star, then watching cat videos on Youtube would be like observing the outermost corona from Earth. Reading the Lainzine and hanging out in #lainchan is like wearing sunglasses while burning your face off against the surface of the Sun.*
> — Anonymous

A vision of the future which brings to life the idea of a harmony between man and machine. No distinctions are made between the AFK world and the Net. A way of living, a way of seeing and learning to swim in these new and exciting times. I give you the Way of Schway.

The Legend of Anonymous and the Lain

The Net has a certain way, a certain unspoken legend, which tells of an Anon. Anon came to the Net a young, innocent child, and he had fun, doing nothing of particular import, and he was nurtured by the bright lights, the moving pictures and the radiant illusions. He discovered the community. The community talked amongst each other, and Anon would participate, and become the authority on such matters, until such a time when he felt the discusisons redundant, and he went on to another place, to again enter the community. Such wise men of the community would say, this leaving and never coming back was the final goal, that one has graduated from the message board. To leave the echo chamber, and to seek out a new one, thus achieving growth.

Many boards exist, to ensnare, filter and catch the various drifting Anons, whose consciousnesses unfold into dazzling networks of connections, before eventually settling into their respective places. For you see, everyone seems to become "stuck", at some point. But, there is a legend, of an Anon, who continued to search. And as Anon journeyed farther on, he began to feel as if he were carrying on the unspoken hopes and dreams of all those whom he had met, because he told himself he would keep walking, and he wouldn't get "stuck". He would walk into a room, learn everything he could, then depart without a word. Always moving, an eternal wanderer. And now, for some such reason, at some place and time between then and now, he has come to Lainchan...

Growing Pains and a Wired Society

Everyone is connected. Prolonged exposure to the Wired, from a young enough age, will reveal this – resulting in a sense of identity with the humanity which underpins much of human consciousness. This is an oft-supposed but mostly undocumented effect that manifests in children who have grown up using the Wired all their lives. In the 80's, this effect was mostly limited to the users of Usenet, but today, the use of Web 2.0 is much more widespread – the private experiences of a chosen few have given way to its widespread use and abuse. Society right now is in transition, as people struggle to understand the full implications of this connection.

Consciousness and the Shell

The similarities between the human body and a computing environment have been often expressed, and we could most simply describe man as a biochemical machine, controlled by an awareness capable of receiving data and issuing commands.

The command is emenated, the kernel receives,

the shell interfaces, and the command is carried out by programs down the chain. It is not the command line itself which creates these commands – rather, the commands are dictated by a higher power, outside the computer's dimension of existence. A user who understands this fundamental relationship and its implications in fullness could be called self-aware.

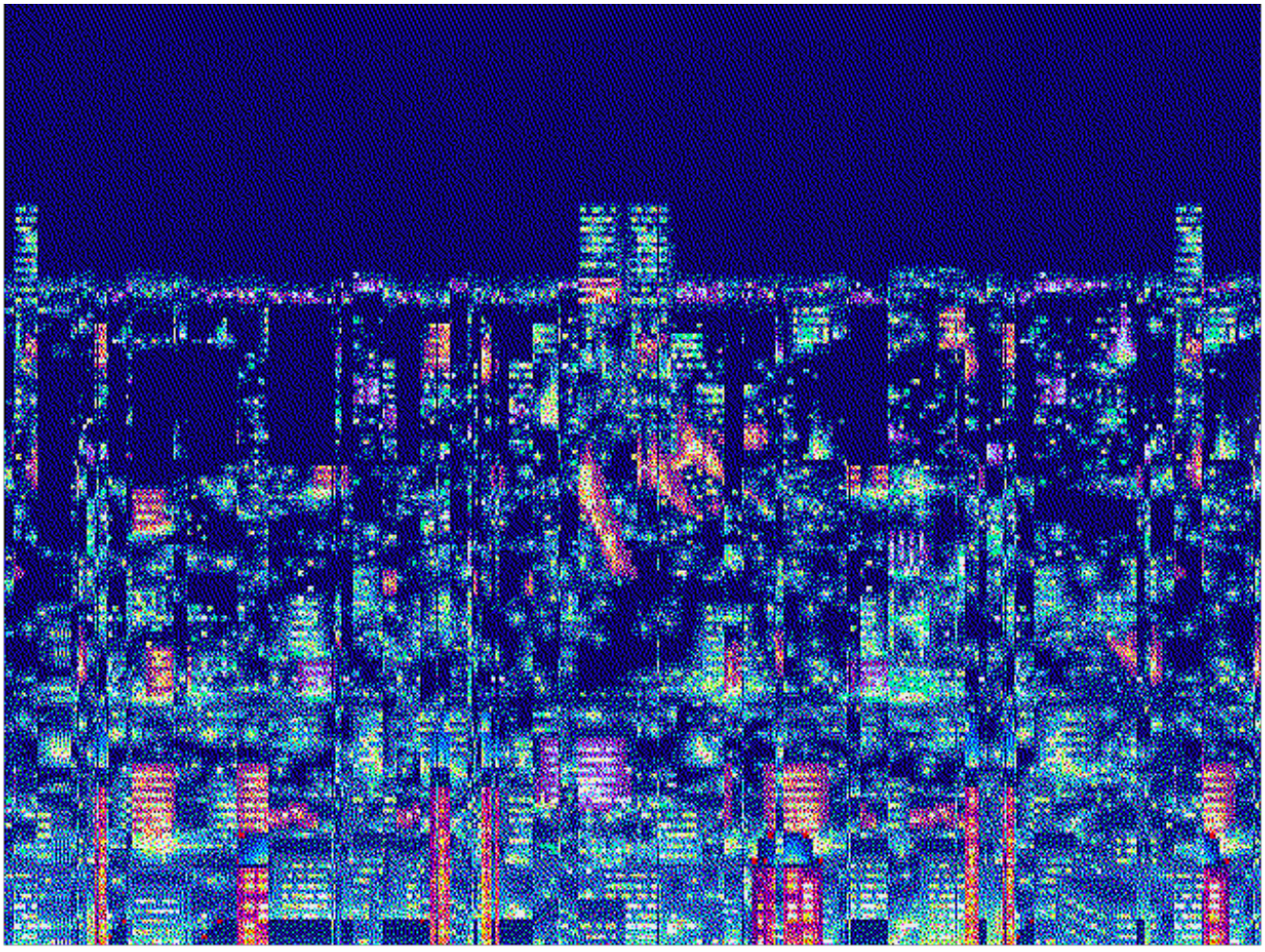
In many proprietary operating systems, self-control is something that is not readily relinquished. Most mainstream operating systems present themselves with the illusion of being easy to use, but become needlessly complicated when the user seeks to use the system for anything other than its “intended purpose”, when the idea of computers themselves having “intended purposes” is completely preposterous. Yet in this way, many users remain unaware of the constructs which silently govern their sessions, never fully grasping the concept of control, and they are thus controlled by the forces which surround them.

Programming a computer is therefore a microcosmic exercise in programming yourself.

Operating Systems

The operating system is much like a set of programs. It can be said to be a person's color, their presence in the world is expressed through their choices, culminating in their intent. It is what determines the context of a person's habits, and grants the vision of continuity to their actions, which would otherwise just be a series of unconnected points in time. It is much like a narrative, a construct, or a computing environment, within which actions are carried out.

Operating systems posess certain attributes, one of which is that they are usually centered around a central purpose, context, or definition of reality, around which the symbols which make up the body of the ego unfold. For example, take Kali Linux – a Linux distribution designed to penetrate network



security systems –, or a particular music scene or style. At their most basic level, they are attributes, constructs, which emenate from one's initial intention, and provide a means of abstraction for carrying out that intention, and coalesce upon the achievement of the intention.

Operating systems need not be installed at once. In fact, a person may employ multiple operating systems to achieve various ends. For instance, Windows may be used to conduct social networking and play games, while Tails OS could be used to exchange secret messages. Similarly, participants in the underground rave scene can use it as a vehicle to achieve certain transcendental experiences. Someone who chooses in the moment to identify as a troll can criticize a certain point, to reveal deeper truths.

It is evident that, by this method of abstraction, there exists the underlying idea that there are infinitely many operating systems which can be created, and that to know the symbols, and to know the packages, is to posess greater power to create one's own system, and in turn, to fulfill one's intentions more directly and eloquently.

It should be noted that a person's intentions are *not* their operating system. Rather, the operating system is an environment most conducive to the intent, with certain systems fitting certain lifestyle models and intentions better than others.

Programs and Daemons

Upon selecting your operating system, you then choose to install programs. You have different types of programs for different things, which run with and without your awareness, in a recursive process. Your current task is the most important thing that should occupy your attention. Outside of that, when idle or between processes, daemons (aka background programs) should be run and allowed time to process, but only as much as is needed to perform maintainance tasks. A focus lies on minimalism – the less you have, the fewer daemons you have to run, and thus the more lightweight your system. Your house, your surroundings, and your life, are much like an operating system on a computer. If you have too much stuff, your actions will be cumbersome, and heavily delayed. Shed the excess, focus on your main process, and experience true eloquence. It is better to simply discard a program you do not need than to let it inhibit your process. Do not become too lost in organizing your system, or become too attached – start with something that works, and maintain your system as you have time to evaluate your surroundings and deepen your understanding.

Understanding and Finding Bottlenecks

Bottlenecks are things that constrain an otherwise productive flow of energy. Poor physical health is a bottleneck – not only are you less able to move around, but it has an effect on blood circulation, and thus mental performance. A job or career you are not satisfied with is a bottleneck – it restricts your free time, which is something you don't have an infinite amount of – thus it is very important to allocate those resources properly. If you were playing a fighting game, and you understood the theory behind how the game is played on the compeitive level, but you didn't have the physical reflexes to seize the advantage, then there's a bottleneck you should undo. The same can be applied through really almost any venture, behavioural, mechanical, mental, or otherwise.

Electricity itself travels extraordinarily fast, and our bodies and minds themselves are the very image of a computer. In computing, if something has a poor write speed, then your transfers would of course, take forever. This is only relevant when performing processes that require a large amount of bandwidth, and for relatively simple tasks, such as the decision-making processes, you need surprisingly little. You do not need as much as you think you need. However, at a certain point, you will come up against these bottlenecks, and as you feel them, you should seek to undo them, to liberate yourself from your circumstances.

The Fountain of Knowledge

It stands to reason that the greatest hindrance on your performance is ignorance, your life span the final, greatest bottleneck. In these moments, you can trust that the Wired will provide (in most cases), the knowledge of any field of science or experience you might need. Although you may not posess the symbols or knowledge now as an individual, it is safe to assume you will in the future, and can therefore say that there is nothing you do not or cannot know – the Wired is much like a facet of your own memory, with other people striking out on the same beaten roads as you or I. That knowledge will flow into you – you need only direct the stream, and tap it when needed. In this sense, it is much like a package repository.

You will, of course, come across paths less travelled, in which no knowledge exists. It is then your duty to break new ground, to share your discoveries with the world! Remember, sharing is caring :^)

The Free as in Freedom Philosophy

A tool that inhibits your ability to function, or restricts your actions in any conceivable form, is something that should be done away with. As someone who seeks to do things, you, of course, want to maximize your diversity of choice and potentiality through the process of liberation. For example, freeing yourself from existing dogmas or ideas, restrictive or coercive situations, and financial restrictions. This is a process, and will not happen overnight.

There also exists the unfortunate reality that many non-free tools and paradigms are unfortunately those which have the most dominance in our society. Thus, it really helps to have two computing environments. The normal, "nonfree" one, in which you conduct your mundane affairs, and the "free" one, in which you conduct your serious business.

You can, of course, have more computers. In fact, you can never have enough computers, your only limit is the awareness to manage them all. You can create computers within other computers in the form of a virtual machine, or a VPS, which can be extraordinarily useful for funneling traffic and creating secure environments. To start with, though, I would advise you to have two computers, your public platform (i.e., Windows, or a smartphone) and add on more platforms, (servers, phones, consoles, VM/VPS) as you learn more and the need for them arises.

Sock Puppets

By successfully compartmentalizing your digital activities into seperate operating systems, you can, for instance, maintain a presence that gives off no outwards appearance of doing anything departed from the norm, while at the same time doing all the things you want to do, fooling any behavioural algorithms that may be looking to analyze your behavioural patterns. There are guides to anonymizing your traffic that can be elsewhere, but the reality is that we are approaching a world where a

high degree of self-control will govern online interactions and serve as the primary barrier of entry to controlling your presence, both the record of you that exists within the NSA databases, and the perceptions of those who know you. You need to control and rewrite those records.

By the same token, you may find it beneficial to adopt alternate personalities towards the fulfillment of various different goals – for example, I am typing this article as "serene", but I use numerous other handles, in different places, all over the Net, and I change them routinely. I only exist as far as this article does, but I nonetheless exist outside of that. A name or a presence is only as useful as the actions attached to it, as a unified symbol. Sometimes, it is better to remain anonymous. Don't limit yourself to a singular identity – grasp anonymity and use it to your advantage.

For instance, take the phenomenon of viral marketing. The central idea is formed. The work is created. Mundane accounts (run by you) from mundane computers (a botnet or series of VPS) spread the work around, each driven to spread the content with a different personality and prose.

A person living a double life at a corporate job. One personality for work, one for play. Those two worlds need not intersect. Don't become too attached to one identity, or it will become harder to dismantle it. Know you are the illusion, but at the same time you are not.

Resource Allocation

An eye should be kept on the objective situation, as well as the subjective. Sometimes a cheap, simple environment is all you really need. Even with bottlenecks, your environment may still work well enough for getting things done. Trust your lust for greater tech, pursuit of greater heights – but only when you've really reached those heights, and find yourself in need. You don't need the newest smartphone. You don't need that home internet connection – learn how to crack networks or go to McDonalds and use some free Wi-Fi. Sever your dependence on corporations – hack something or find your own way to communicate.[1]

That's all for now. Tune in on next issue for social engineering and blackbox abstraction!

❧

1 I'll just leave this here: https://www.youtube.com/watch?v=SBPvBLZcx9c

Repairing Old Electronics

This guide is for people interested in repairing and troubleshooting their devices. You do not need a background in electronics, this guide covers the basics. (Please note that some of the equipment mentioned here is not necessary depending on what you're repairing.)

Must have: soldering iron, soldering tin, screwdriver set, rubbing alcohol, cotton swab (Q-tip), and some needle nose pliers.

Nice to have: solder pump, megger/megaohmeter, multimeter, heat gun, shrink wrap. Magnifying glass (little helper).

Why you should consider repairing old electronics: The answer differs from person to person. Some buy old and broken electronics, fix them, and sell at a profit. I do it to preserve my old equipment. Others do it to learn or simply for the fun and challenge. If you can't think of any reason why you'd repair old electronics, then this guide is probably not for you.

Okay, so let's get to it: Go find something old and simple like an amplifier/radio, LCD TV, Commodore 64, or a Gameboy. Do a visual inspection of the board. Are there any obvious burn marks? Has the board been tinkered with before? Do traces on the board cross? The most common items to find fried on a board are the resistors[2]. You want to check the color coding used on the fried resistor so you can figure out what to order. There are a few phone apps you can use to find color codes. I recommend buying bulk buy resistors, since they are very cheap on the wired. Blown or bulging capacitors[3] quite often leave a device unable to power on. You can tell if a capacitor is damaged by looking at the top (and the bottom if possible). A good cap should be flat in those areas.

If the board is dirty, clean it! Dirty boards are not fun to work with. Blow dust off the board, and clean the rest of the dust off with a cotton swab (or several). If the board is sticky, apply rubbing alcohol on a piece of paper or cotton swab and clean it throughly. A lot of old cheap solder secretes a gross oily looking substance. If you see corrosion[4], use a brush (a old toothbrush work fine), some rubbing alcohol and start carefully scraping away the corrosion. Corrosion can come from bad capacitors, old solder, and dirty water.

Turn on the device. Does it turn on but doesn't do what its supposed to? Learning the right questions to ask yourself before tearing into something can save a lot of time as most electronic failures aren't hard to fix. Sometimes old equipment gets thrown out because of water damage, and water damage isn't as bad as it sounds in most cases, generally it should work if you leave it off for a few days to dry. If it doesn't turn on, check the power button and fuses (if you don't know what they look like, look up “2A fuses”). Are the fuses intact? Sometimes it's hard to tell if a fuse is blown or it isn't transparent. Check for blown/bulging capacitors. It is a good idea to have a few known working fuses around to test. Does the power button properly work? Sometimes old solder cracks or falls off when it gets too hot, A few minutes with a heat gun is good at fixing this. If the fuses are intact, no wires are obviously disconnected inside, and if none of the capacitors seem damaged the device might require further troubleshooting.

To find out if the board has been tinkered with before, look for wires going from one part of the PCB to other parts of the PCB, or the manufacturer date on capacitors and other parts of the board. These jobs are sometimes done badly. Is there too little soldering tin? Are the wires loose? Did the previous hacker use enough tape or too much super glue? (too much glue can lead to overheating and a blown capacitor). Are there wires twisted together where they should be soldered?

If traces are crossing, then you probably just found the issue of the board you're fixing. If it is

2 http://images.wisegeek.com/tan-fixed-resistors.jpg

3 What a healthy capacitor looks like: http://s.hswstatic.com/gif/capacitor-1.jpg

4 https://commons.wikimedia.org/wiki/File:PCB_corrosion.jpg

soldering tin thats crossing the traces, then consider using the soldering iron to heat up the tin and remove it with a solder pump, followed by covering up the area with glue. If the traces are missing however, then you might want to bridge the traces with a small wire. (If you don't have any, you can split open a unused Cat5e (Ethernet) cable and use them for bridging. Be warned, it can be a tedious job to solder such a small wire).

If you can't find any obvious visual problems on the board, then the multimeter and megger can come in handy. Every multimeter is different, but they come with some basic utilities such as checking voltage, amperage, continuity and a continuity beeper. We will focus on the continuity beeper, a great tool for checking if there are trace connections between the components.

You want to first of all check the continuity from positive (+) to negative (–), + to ground and – to earth with the megger. If it displays less than a megaohm, then there is a shortage and possible traces crossing. Do another visual check because you probably missed something. If the megger shows 1 megaohm or greater, then there are no traces crossing and you may proceed to use the continuity beeper between capacitors and other parts of the board. If you find one that does not beep, you might have found the issue.

Note: Sometimes the fluke can damage your equipment due to overcharging. Depending on how much the board can handle, you can also use the continuity beeper for checking +, –, and earth.

If you still haven't found your problem, look up the product name and “common problems”. Sometimes electronic devices has common problems degrading or breaking a product, there should be documentation online on what the issue is and how to fix it.

Resources

- https://www.youtube.com/user/EEVblog: Great lessons on the different tools covered in this guide, as well as fundementals of electronics.
- https://www.youtube.com/user/lukemorse1: Repairs arcade PCB's, great for learning more about troubleshooting electronics.

A Dream of Lain

The following is a dream that recurred several times throughout 2013. I have many records of it, so I can piece the forgotten parts together. Lain is quite out of character in this dream, but I feel the need to commit it to words.

I was walking through a street in Japan with some friends. I don't remember anything about the friends, apart from that I felt nervous with them. We came to the brow of a hill, and people started to disappear. I looked around, and the dream cut away to the inside of a little suburban house. It was filled with acrid smoke, like plastic or rubber burning.

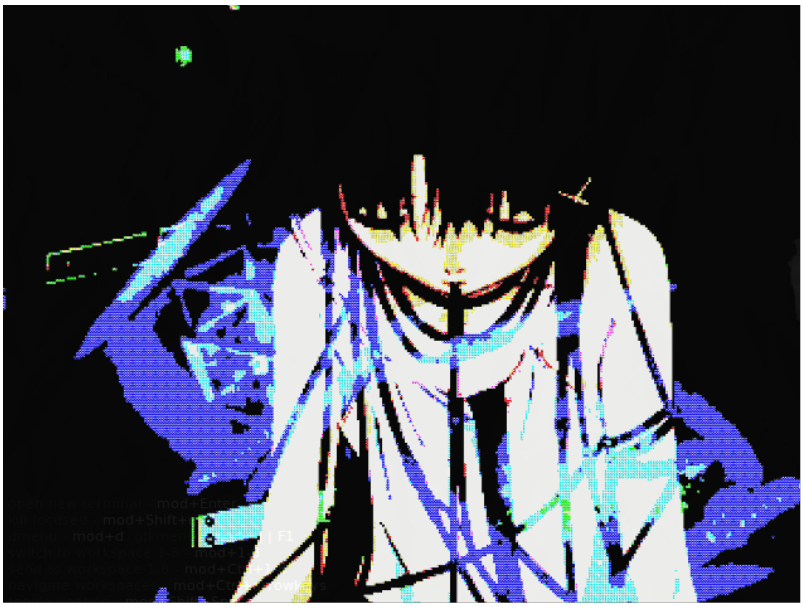
In the house, people with little pink hollows where they should have had eyes roamed the corridors, stumbling, flailing. Lights flashed. I walked through a sort of rubber corridor and met a beautiful girl.

The girl was in her early teens. She wore nothing but a white nightdress, and the cross-shaped hairclip in her curious length of hair that escaped her hacky, DIY haircut. I knew immediately that the girl had been carrying out experimentation on those unfortunate enough to wander into the woods (I have forgotten how I knew in the dream, sorry). Nonetheless, she was beautiful.

The girl confided in me. She told me that she was working to build herself a father. She had turned away from the world before, but she always watched from the bridge over the underpass by the school, watching all the people who don't know her name.

I felt calm. Wonderfully, suffocatingly calm. The girl laid me down on a cold metal bed and asked me to remove my clothes. She explained to me the human need for permeation, saturation, penetration. She then drew a neat line from the top of my forehead to my clitoris with a knife. It didn't hurt; my only thought was that I would be in trouble with my mother if a scar carried across to the real world. I was aware that I was dreaming, but not quite aware that what was happening was not quite real.

The girl opened me up and slipped inside. I lay, happy in the thought that I was filled with her, that my blood and organs and pallid brain were keeping her warm and safe. I knew now that it was certainly a dream, and I thought back to my days of paranoid delusions, when I would be caught by mother babbling about the CIA in my sleep. I worried that perhaps I was making sounds in my sleep. But the dream didn't end.



The girl spoke to me, resting her brain against mine with a tingling warmth. I can't remember the words, but they were like sweet nepenthe to me. The girl had made so many people suffer what looked like unimaginable pain, but she made me feel special. Perhaps she had given them what they truly needed, too, and it simply looked painful.

I woke up, but rolled over, fell back to sleep, and immediately re-entered the dream. We stood up and walked around the room, the girl all the while speaking to me. I woke for the last time, got up, dressed, and went to school.

❧

Recommended Reading

'Programming from the Ground Up'
by Jonathan Bartlett

by FORMAT

The full book is available under the GFDL at:
https://savannah.nongnu.org/projects/pgubook/

This recommendation is suited for any reader looking to learn how a computer actually works. This is a book that teaches programming a little differently than most other books, but still manages to be a worthwhile read for both experienced and upcoming programmers alike. Most programming material is `top to bottom' in terms of the abstractions taught; a novice is typically expected to learn abstractions and only learn how they work at a much later date. Books such as this one challenge that idea by working in the opposite direction. Programming basics are taught at the low level and high level languages are discussed later.

Topics covered include:

- basic computer architecture and terminology, including processors, memory, and addressing modes,
- assemblers,
- algorithmic problem solving,
- function calling conventions,
- recursion,
- basic file I/O,
- basic error handling,
- libraries and dynamic linking,
- basic memory management,
- computer numbering systems,
- program optimization,
- system calls,
- assembler language idioms,
- a short introduction to using GDB.

In closing, this book is a valuable resource for anyone looking to break into the daunting world of assembler languages. GNU tools are used, along with AT&T syntax; I'll simply say that this is a plus for anyone who is confused. The x86 instruction set is used for this book, which is par for the course with books on this subject. For experienced assembler language programmers, this book may prove a nice resource. Regardless of skill level, I think you'll enjoy reading it.

It's recommended that you download the source for this work as well, to have easier access to the example programs.

Untitled

Once upon a time, I wanted to play *Grand Theft Auto 3* like everyone in the neighbourhood, but the parents said no. Later I rode ATVs because they weren't regulated like automobiles were. It's really unfortunate to live in a country where most everything fun has age restrictions. I notice many unoccupied campers, shacks and cabins along the roads and trails I ride. I mention this to some girls I know who ride. They say they broke into a camper recently, it was easy. I'm intrigued, this sounds fun.

We go out and stop at this camper near an old falling apart building. I look around. There appears to be no-one around. I check the door and it's locked. The door and lock look flimsy, with a strong pull they give way and the door's open. Once inside, I look for valuables only to find a bic lighter and some coins. One is a state quarter. Free money! This is too easy. I grab a paper towel off a roll hanging up and wipe down where I touched, only to get laughed at. The police don't put much work into investigations, I'm told.

I find this game fun and exciting. We continue to do small acts of theft after school. Not much is gained because people tend to keep valuable stuff elsewhere, in houses and sometimes cabins. I suggest to my friends that they try some cabins. They all agree. I feel we should get some gear togeth-

er, maybe because I've read the anarchist cookbook lately. I find an old backpack for school and put inside a hacksaw suitable for cutting chains, a crowbar for prying open doors and a maglite to see, maybe hit people with, and lastly a .22 pistol that my parents left laying around.

Friends and I set out to rob a proper cabin. One of the friends suggests one that looked promising, but it's watched over by neighbors. I don't care, we can just bring more guns after all. That's what you see on TV, right? Everyone rolls in and I hand a .380 to one of my friends, tells] her to be a lookout. I grab a crowbar and head to the door. This lock can be unscrewed, my friend says. So they get out a tool kit and take off the padlock. Inside, it's a fucking goldmine. I start filling my bag with all kinds of ammo for guns I couldn't find (sadly), and a Schrade lockback knife left in the kitchen. A friend tells me there's a gun. I'm excited, but sadly its an air rifle. Time to go!

Outside, a friend says she doesn't know how to unload this gun and maybe she should shoot it. I think this will attract attention. Everyone leaves, but partway down the road we run out of fuel. A friend remembers a sawmill where fuel is kept. We all go to liberate it and after vandalising the sawmill we leave with a container of gasoline. Once refueled, we head home. I look over all the loot. So many bullets for so many guns... One of the mags even works in a rifle of mine.

The next day, I go back and search the cabin. Nothing new, but I take the air rifle. Leaving, I notice a deputy sheriff. I remember the loaded .380 pistol in my pocket and knows if I'm caught with that its juvie... So I drive away full speed and head off road. I ride to another county and wait many hours before returning home another way.

My friends think this is bad, but we are not deterred. We set out the next day to rob a house. We find a nice one, out of sight from people that can call the cops. The door is dead bolted and the crowbar just won't budge it. I hold up a friend so she can

kick in a glass window. These people don't have much worth taking, save 2 boxes of federal shotgun shells. But wait, what's this red cylinder? A fire extinguisher! We spray and trash the house. This is so much fun.

Outside, there was an outer building with a gun visible through the window. The door doesn't open, no matter how much I push on the crowbar. So I sit down and think. What if we use the ATV to ram the door? 2 minutes later, the door is open but it's another damn air rifle. Everyone leaves.

Another time, we travel to to this area with a bunch of mobile homes. It seems everyone is away, but the place is gated. I start sawing on the gate's chain, but a friend shows that there's a hill we could ride through and gain access. We enter and there's like 8 mobile homes, so I set upon the doors with a crowbar. There isn't much in them but some kind of toy gun, clothes and boring shit. I spot a shelter with a locked cabinet. Everyone goes over there and it's padlocked. I also spot a box of playboy magazines, which will be fun to take after the cabinet is opened. Crowbar is useless, but I spot a maul, which is like a combination axe and sledgehammer. 20 loud as fuck blows or so later, the lock is off and I'm going through the cabinet. It only has an ammo box full of handtools and other boring shit. Then friend says she hears someone coming. We argue then spot a truck coming towards us. OH SHIT OH SHIT OH SHIT

Everyone hops on the ATV and take off with a truck in pursuit. I remember the .22 pistol in my backpack and take it out. I aim it at the truck but there wasn't a round chambered. We slow down as the bolt gets pulled, but by then the threat is away so I put the gun away. Friend takes off her hoody and tosses it off a hill on the way home. Once safe, friend goes for a bath while I think about how fucking close that was, my adrenaline still pumping. I hear my name called and I'm asked through the bathroom door to go back and get the hoody, since it had a school logo and some personal things. I laugh and say, sucks to be you, get it yourself. She says open the door, and I do. She's naked in the bath. Please go get it or else I might get in trouble. I suggest she doesn't want to and friend says come on you are getting to see my boobs. I don't think C cups with tan lines looks so great, but agree to. I take a walk through light sprinkling rain because anything else seems too sketchy. Hoody is found and a long walk home ensues. My friend is happy to have it. She leaves.

Next time the friend and I go to another cabin since they are safer to hit; after breaking a window on the front door we are in. Only this isn't a normal cabin. It has weird shit in it. Like a second half of a deer mounted to the wall with a toy vagina attached. Everyone is sketched out. I suggest we burn the place to the ground but no one feels like going to get a can of gas and a lighter so everyone just wrecks the place and writes comments on the wall about how sick it is. I'm tempted to go back for a pic. On the way home, mother waits for me and asks why I have a backpack. I don't really have an answer, so mother looks inside and snaps when she finds the gun. I get in trouble, losing my guns and crowbar.

That weekend, I visit a gun show and trade stuff for a dummy pineapple grenade and a .25 automatic pistol. My family tells me the area they work in is

heavily monitored, so I suggest we switch to home invasions. One friend agrees, but the other says no. Later we set out to rob a house with someone in it by an area that's pretty much been robbed clean. I suggest taking the .25 but I'm told it's not really needed. So I grab a looted kitchen knife that looks mean and pocket the hand grenade. We set out to some houses but notice people by the road. What a bunch of stupid dumb rednecks, I thought as one jumped on the ATV, turned it off, and took out the key. I wish I still had that gun so I could shoot all 3 of them, but I'm too sketched out to even pull a knife. Friend talks them into believing everyone was innocent and somehow know one of them. Everyone is let go and ride home but we notice some others with a similar ATV and one looks like the girl I'm with. I suggest we warn them, but friend is like fuck that, let's get out of here.

I go home to learn someone had called her parents and told about the stop. Everything got downplayed by the lynch mob because they thought my friend and I were innocent. Dumb fucks. I decide to quite whilst ahead. Later, I read in the newspaper that some poor sod confessed to everything I did and some other shit I didn't know of.

A few years later, a classmate told me that some asshole robbed their mobile home half an hour after getting caught having sex on their picknick table. They had an ATV just like mine, only it was 4x4. That's how they knew it wasn't me. I agreed. I felt sorry for the guy in jail who was mostly innocent and decided to never do shit like that again.

The end.

❧

A Crash Course to LaTeX

Do you ever get annoyed while writing papers because of a lack of layout consistency? Do you hate it when you need to change everything by hand? Do you ever try to click on something that just won't highlight, for some godforsaken reason? Or want to display fancy mathematical formulas, but your WYSIWYG editor is too cumbersome to handle?

If you answered yes to any of the questions above, I've got just the thing for

you: LaTeX. Now, before you put on that catsuit, let me explain to you what LaTeX is.

LaTeX is a markup language designed to be both logical and easily extended. Most of you probably have at least a little experience with markup languages: Markdown is a very popular one, for instance. But why would you use LaTeX over any other WYSIWYG editor? It's easy: because it's easier to keep track of your document's structure, less hassle (usually), and because it has implementations for lots of neat formulas. But let's cut the crap, and start editing.

Installation

Before you begin, you should know that LaTeX is written in plain text (saved with the .tex extension), and then converted by a special program, to all sorts of formats (for example, HTML or PDF). On Linux, you can install a package called `texlive` with your favorite package manager, and then execute `pdflatex <filename>` to convert your file to PDF. I don't use Windows or Mac myself, but MiKTeX is popular for Windows, and MacTeX is popular for Mac, so those are probably your best shot.

Besides PDF, there are a lot of other formats LaTeX converts to, like HTML, DVI, PostScript, RTF, and even images, for some reason. These all have their own compilers, which you can also easily find with a quick search on the wired.

Actually Starting

Alright then, now that you've got your LaTeX-to-PDF

converter all set up, let's have a look at some text you can compile, shall we? I believe in learning by example, so that's exactly what we're gonna do. Now, LaTeX uses *commands* to perform markup operations; all commands start with a backslash, and look something like `\command[option1][option2]{argument1}{argument2}`.

Generally speaking, most everyday commands only have one argument and no options, but there are more complex exceptions. The syntax is always the same.

Here is an example LaTeX file:

```
1  \documentclass[a4paper,12pt]{article}
2
3  \title{Chocolate Mooose}
4  \date{}
5  \author{The Swedish Chef}
6
7  \begin{document}
8    \pagenumbering{gobble}
9    \maketitle
10   \newpage
11   \pagenumbering{arabic}
12
13   \section{Step oone}
14     Noo, toodie wee well meg dee \emph{chocolate mooose}.
15     Step oone, geet a five-pound block oof chocolate.
16
17   \section{Step Twoo}
18     Step twoo, get dee mooose.  Heer, mooosee mooosee mooosee
19     mooosee!
20
21   \section{Step three}
22     Step three, put dee mooose een dee bleender.
23
24     \textbf{Nuutice}: dee bleender cun't hundel dee entlers.
25     Put dee entlers een last.
26
27     \subsection{tiip}
28       Seeve dee entlers, yoo cun use dem fur furks.
29
30   \section{Steep fuur}
31     Steep fuur, put chocolate een weeth moose, und bleend egen.
32     % This is a comment. Hi.
33
34   \section{Duuuune!}
35     Uum!  Dees ees guud mooosee!
36 \end{document}
```

Decyphering

Now, there's a lot going on here, so we'll go through it line by line.

1 In line one, we define the document class, which defines how our article will look. Our document uses "article", which is the most frequently used class, but many others exist. Feel free to look into these, but if you ever don't know what to pick, pick article. We also tell the compiler the size of our paper (A4) and the size of our font (12 pt).

3 to 5 In these lines, we define the author, date and title for our document, which will be used on our title page. Since I haven't the faintest when this sketch first aired or was written, I'm leaving the date blank. If you completely leave out the date command, it will default to the current date.

7 and 36 These lines declare the start and end of our document. The actual formatting will be done between them.

8 This command makes sure that the current page (the first one, in this case) gets excluded from the page numbers. We want this here because this is our title page.

9 This command formats our title page with the title and the author. Normally the date would be displayed here too, but since we left that empty, it won't be.

10 This command makes sure the rest of the current page will be kept blank, and puts the 'cursor' on the next page.

11 Here, we set the page numbering to `arabic` (1, 2, 3, 4). Other possible arguments are `roman` (i, ii, iii, iv), `Roman` (I, II, III, IV), and `alph` (a, b, c, d). Small roman numbers look like soykaf though, don't use them.

13, 17, 21, 30, and 34 These commands declare, as you might have guessed, sections. A section is a part of the document, with a title above it, which is specified in its argument.

14 The `emph` command in this section puts emphasis on its argument: normally this displays as italic text, except when you nest emph commands, in which case the second emph will become normal text again (so it still stands out).
24 The `textbf` command here makes text bold.
27 This command declares, yes, you've guessed it once again, a subsection. These get slightly smaller titles than normal sections.
32 The text after `%` is a comment. It will be ignored by the compiler. Comments can be placed in-line as well as on their own lines.

As you can see, the actual text in every section is indented. This is not required (at least not with most compilers), but considered a good organization practice, as it makes your document easier to read and edit.

Conclusion

This article was but a short introduction to LaTeX. If you want more (and you should), I highly recommend the wikibook listed in the credits below.

LaTeX might feel a bit alien in the beginning, but it is really the soykaf once you get the hang of it.

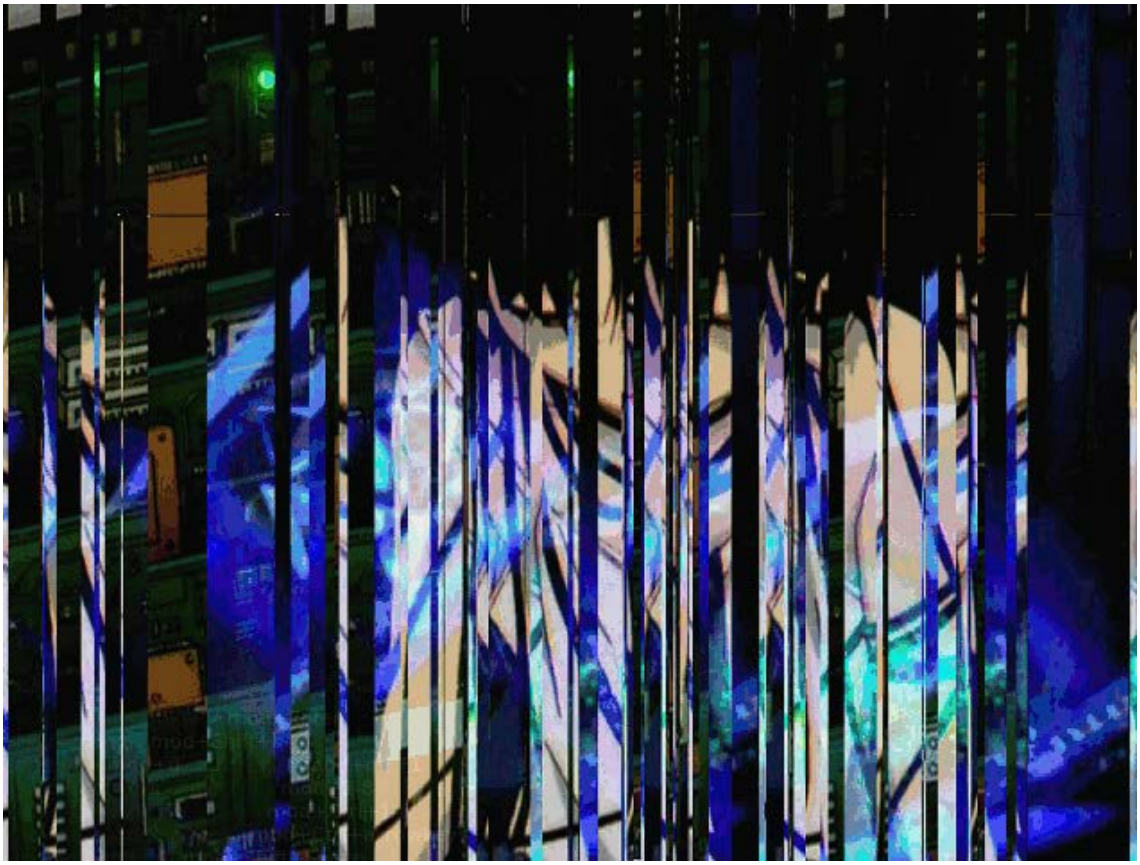
Thanks for reading.

The source file for this document and the example, if you want to check them out, can be found at https://vivesce.re/lainzine/latex.

Credits

- http://www.puppetresources.com/documents/script49.txt: Place where I got the muppet script.
- https://en.wikibooks.org/wiki/LaTeX: A wikibook about LaTeX, from which I pulled some information I had forgotten.
- LaTeX: Some silly markup language I wrote this article in.
- My parents: For telling me to not go on nefarious websites with hackers and stuff. I– I'm sorry.

❧



Console Hacking

> You've got all these computers,
> do something cool with them

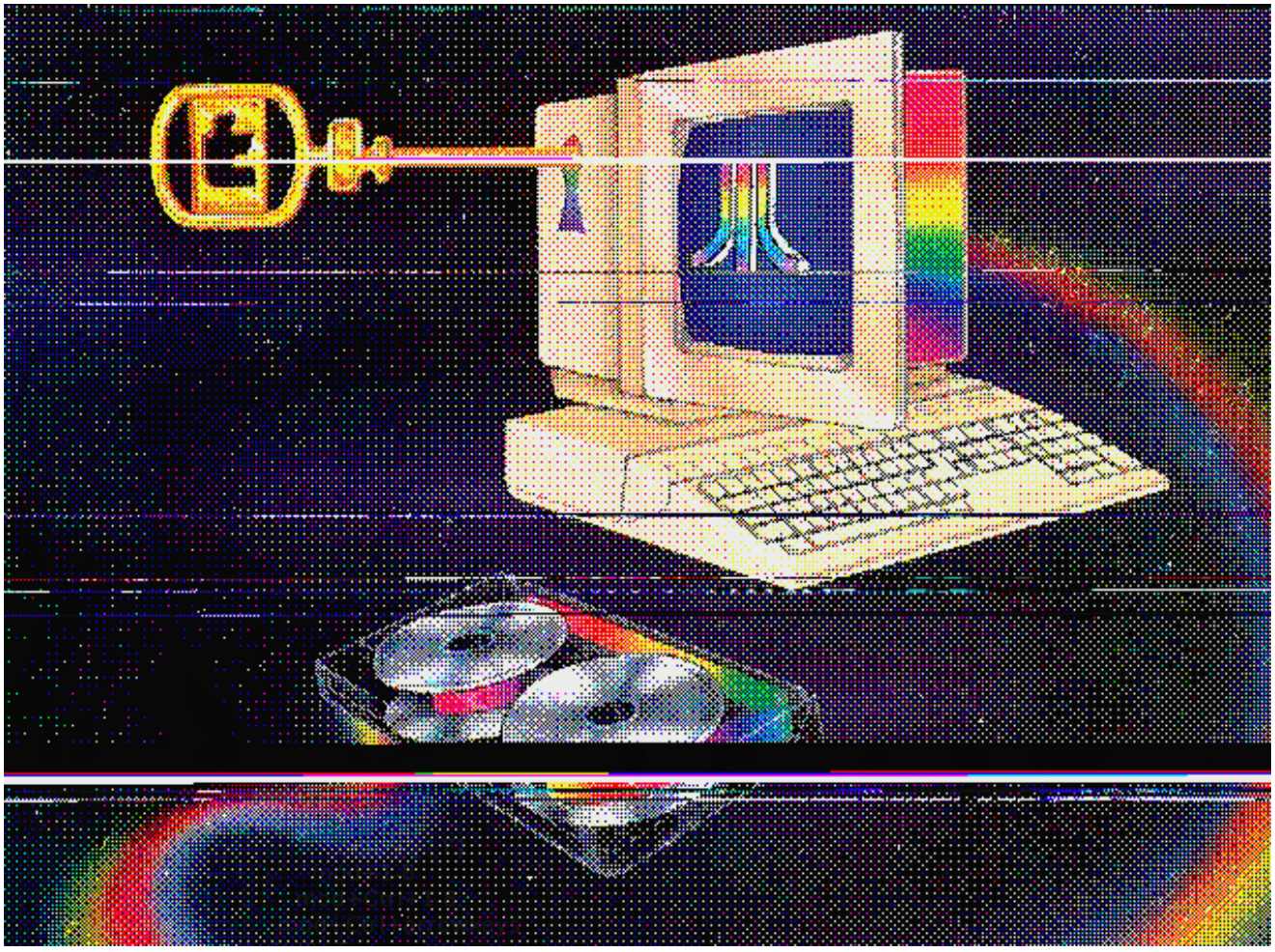
Most of us have at least some form of gaming device, be it a dedicated console or a handheld. More often than not, it's been locked down by its manufacturer, to prevent the user from using it as what it is – a computer. There's a rich community built around homebrewing, or breaking those boundaries, giving fun new uses for your old (and newer) systems, and the main draw: piracy! This article intends to be a cursory overview of some of the coolest things you can do. These are just starting points – I suggest going to gbatemp.net to learn more about how each system works, and as always, Google is your friend. Think creatively! I have a friend who used a Nintendo DS as an offline PDA and a VOIP phone. Why buy a Palm Pilot when you can use DSOrganize?

Nintendo DS

Using a DS for homebrew is as easy as buying a Chinese flash cartridge off the net, meaning it's pretty much the easiest system to tinker around with.

Flash cards – available in two varieties, Slot-1 and Slot-2 (for original and DS Lite). Use them to play ROMs and run homebrew applications. Because of their comparability and extended features, I recommend the Supercard DS One for Slot-1 (DS mode), and the Supercard CF/Mini SD for Slot-2 (for GBA mode), the reason being that the DS One can use the CF as a RAM expansion, and serves as a No-Pass. Supercards have the most features and widest compatability – get one. Other options are the R4 and CycloDS, but they're not as cool.

Passme/NoPass is the name of the protocol which runs off a Slot-1 card, to boot homebrew from Slot-2. If you want to use a Slot-2 card, you'll first need a Slot-1 that supports booting, such as an R4, a Superkey, or any Slot-1 Supercard. Fun



fact: Action Replay DS has a hidden key command at boot to launch NoPass.

Flashme[5] – use a Passme/NoPass card to install custom firmware to the DS. Lets you launch Slot-2 carts from the main menu, and speeds up your boot time.

GBAccelerator DS[6] – a small modchip to overclock your DS, for shits and giggles.

Recommended applications – Moonshell media player, DSOrganize, DSLinux, lameboy, snes9x, SvSIP (turn your DS into a VOIP phone!).

PSP

Hacking a PSP is a bit harder, the idea is to flash custom firmware in order to load homebrew. Your success will vary depending on the model you have. In general, the original 1000-series is the easiest to modify, followed by early 2000 models, which vary based upon the motherboard inside. Follow this guide[7] as your choice of firmware and model will vary based on your circumstance.

There are a ton of well-made emulators for the PSP, making it one of the best portable emulation platforms out there. Some suggestions: Snes9x, gpSP kai, PSPKVM. There's also a Cave Story port.

Playing PS1 games. Because the PSP has a built-in PS1 emulator, you can feed it PS1 ISOs with varying results. There are a lot of PS1 games you can run flawlessly on the PSP which haven't been officially ported, see this guide[8] for more info or grab some of the PSX2PSP EBOOT files off emuparadise.org.

3DS

There are a few flash cards out for the 3DS, the AceKard and Sky3ds for example, but the leader is definitely Gateway. You can play 3DS and DS ROMs off it, as well as run basic programs to manipulate the emuNAND and system firmware. The scene's still evolving, so there aren't as many good homebrew applications out. However, many DS homebrew applications will also run on the 3DS in DS mode, and it's hard to beat free stuff.

At the time of writing, systems with the newest firmware will need to downgrade using the exploit found in Cubic Ninja to use Gateway 3.1, either through the physical game, or a ROM on a Sky3DS. Still, I think the 3DS has a promising future, both as an emulation platform, and also as an NFC reader (in the new 3DS/XL models).

Wii

Softmodding a Wii is pretty damn easy, but it can get surprisingly complex once you really start messing with the internals, and it's easy to wind up with a brick. Ask yourself what you want to do, and follow some of the tutorials on GBATemp to help guide you through the process of getting there. Most Wii's nowadays will use System Menu version 4.3, which requires an exploit to load unsigned code, and install the Homebrew Channel from there. I suggest using Letterbomb to load the Hackmii installer.

After that, you'll want to enable fake signing homebrew, ala the Trucha bug. To do this, you need to install some form of custom IOS, such as d2x. After that, the gate lies open...

Some homebrew highlights: DVDX (lets your Wii play DVD movies), USB Loader GX, Triiforce (for installing channels to flash drive emuNAND), Project M (awesome Smash Bros mod), Wii Web Server, WiiBrator (I'm sure you'll find a use), Ocarina/GeckoOS (cheating in games), DIOSMIOS (load Gamecube games off an SD card). There's also a slew of emulators for just about any system you could ever want – the Wii is definitely the go-to emulation center.

---

5 home.comcast.net/~olimar/flashme/

6 www.division-6.com/products/gbaccelerator-ds.php

7 gbatemp.net/threads/psp-hacking-modding-f-a-q-start-here.268289/

8 www.psxpsp.com/psx-eboot-creator.php

PS3

Hacking a PS3 is problematic, because you'll need one with a firmware of version 3.55 or lower. There's a good chance that any one you buy at the store or anywhere else will have been updated to a newer one. Unless you want to spend money on an E3 Flasher, or pay someone to downgrade an existing console, I suggest you buy a fresh 3.55 system off Ebay, and install custom firmware from there. Still, if you get it running, then you'll have a decently powerful device under your control, which makes a great server or media center.

Fun things to do. Mine bitcoins with cellminer.[9] Get rid of your cable and/or Netflix subscriptions, and stream anime and movies with Showtime through Navi-X. Play Metal Gear Online.[10] Install Multiman, run FTP and web servers, and play pirated games.

❧

9 github.com/verement/cellminer

10 savemgo.com/forums/viewtopic.php?f=9&t=783

Night Ops

A guide to shenanigans and fun things to do when the world's asleep

Remember the cardinal rules of night ops:

1. Only play when you know you'll succeed.
2. Only carry what you need. Don't carry ID.
3. If operating with friends, practice easy things as a group before doing risky things.
4. When in doubt, abort the mission and go home.
5. If you're in danger, drop whatever you're holding and run.
6. Prepare before you leave, plan what you're doing and don't leave home without your brain.

So, you want to do fun things at night of questionable legality? Perhaps you're an aspiring pentester, and you want to get some experience in infiltrating past people, or maybe you're going to try some urban exploration. Maybe you played a lot of Metal Gear and now you're wondering if it works in real life. Or maybe you're just bored, and you just want to break stuff or steal things, to each their own.

When venturing out at night, it's important to recognize that you're fundamentally putting yourself at risk, especially if what you're doing is illegal. There are all kinds of things that could happen, you could be harassed by police, you could be jumped by thieves, or if you're doing some kind of stealth mission, you could be discovered. As such, it's important to take what you're doing seriously, and to not leave home without your brain.

My friends and I have been practicing night ops for several years. This document is a rough outline of how we organize ourselves and how we handle ourselves during a mission.

Missions are generally assigned ranks based on their difficulty and threat model – the highest being an S-Rank mission, the lowest being a D-Rank. People without experience should only attempt lower-ranked missions, until they understand how to function as a team. Before embarking on a mission, ask yourself these questions: "Do I understand all the forces at play in what I am attempting?" "Can the people in my team handle the mission?" "Do I have all the supplies and information I need?". If you can say "yes" to all of those questions, then you can proceed.

Without further ado, here are some suggestions for fun missions you could try, or you could make your own!

D-Rank Missions

Play the "Cars" game. Get from point A to point B without being seen. This includes people on the street, people in houses, and people in cars. Wear dark clothing and try for absolute stealth. If you are seen three times, the game is over, stop sneaking and go home, because it's possible that one of those three people might have called the police on you. This makes a good team-building exercise with a relatively low risk.

Go dumpster diving. Wear thick jeans and boots, carry a flashlight and a small razor for slicing bags. The proper posture for getting stuff out of a dumpster is *not* to dive inside, rather lean in on your waist, like a teeter-totter. The best places to hit are bakeries and supermarkets for food, or tech outlets for things like cables, and the occasional TV that you might be able to fix.

Relatively low-risk. If you're stopped by police, drop your razor and light in the bin, and say that you're moving and were just gathering boxes. Some dumpsters may be locked, requiring some lockpicking skills.

Vandalize something. Hate someone? Pour sugar in their gas tank, or key their car. Smash pumpkins on Halloween. Place fireworks in things you want to blow up. The possibilities are endless, just don't get caught.

C-Rank Missions

Urban exploration. This will probably require some lockpicking skills – learn how and practice before attempting this. Some vacation hotspots:

- the tallest building in your town,
- underground tunnels,
- rooftops of apartment buildings,
- university campuses,
- subway systems,
- abandoned buildings,
- warehouses and factories,
- construction sites.

You may encounter squatters, police, or other unruly people when exploring, so always dress as if you belong in the place you're exploring. A reflective vest or a lanyard and badge can give the psychological effect that you might have some sort of reason to be where you are, even if you don't.

Abandoned buildings are good places to practice graffiti.

B-Rank Missions

Make a drug deal. Take a friend with you and head downtown. Try to score your drug of choice from people who look like they might be dealers. Be wary of people who might try to lure you and jump you, never give someone money before you see the product. Look out for cops, it helps to recon the area a bit before approaching anyone to observe people's behaviours, and sniff out any would-be undercover stings. Definitely only to be attempted with someone you trust at your side, who won't leave you hanging in a tense situation.

A-Rank Missions

Graffiti a public place. Dress inconspicuously, and decide what colors and tools you're going to use before you head out. Travel light, and don't be afraid to split the work into multiple nights, if you think you can't carry everything in one pass, or if

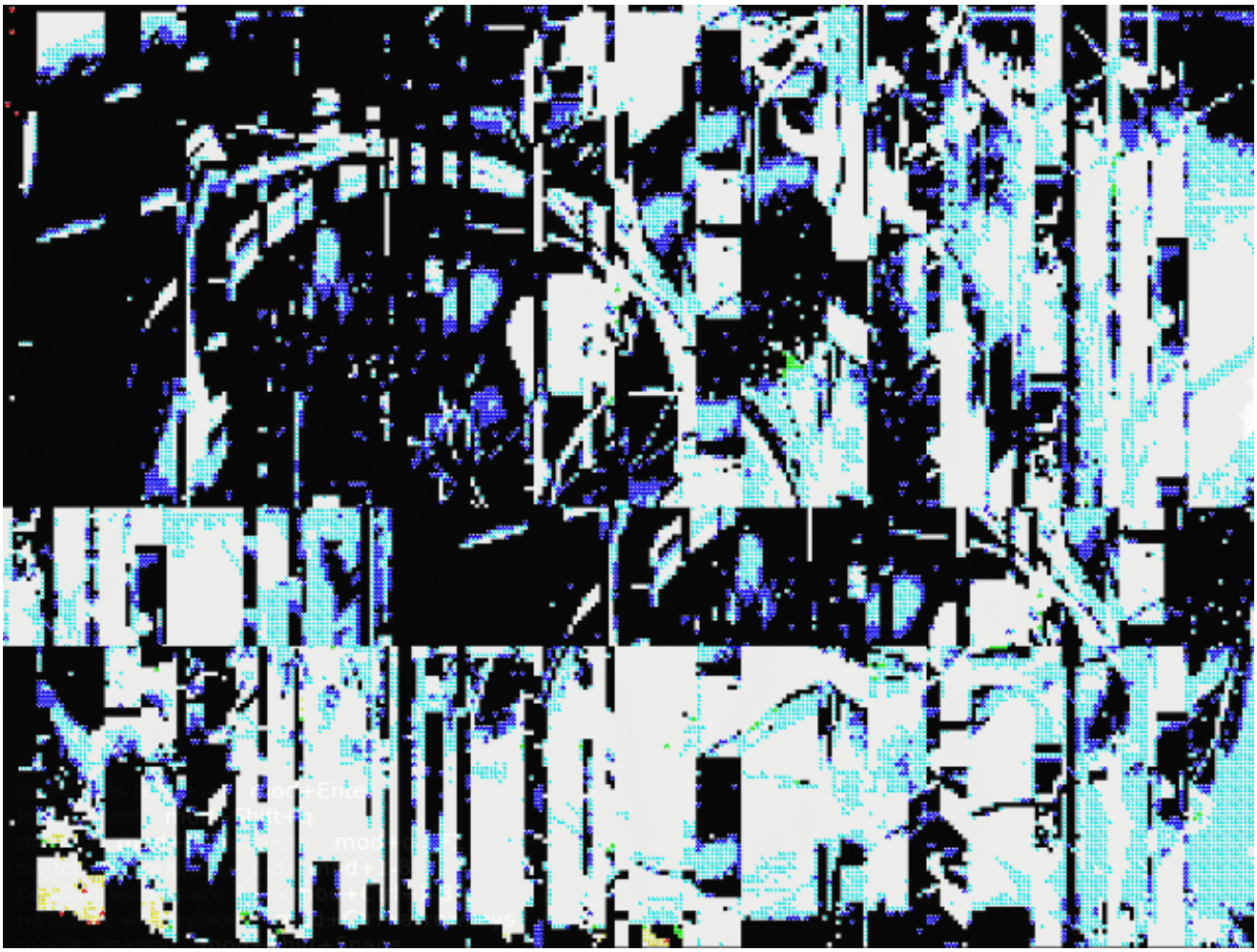
you think you can't finish by morning. If you can, try to conceal your paint, in a bag, or possibly putting that spray can in a chip bag or Pringles tube. It can help to bring a friend with you to serve as a lookout.

If you're bombing a trainyard, and have to paint near tracks, be forewarned of the "suction" effect that can come from passing trains – if you hear a train coming, get as far away from the track as possible, and lay down/hold on to something if you can.

S-Rank Missions

Steal something important, take revenge, or otherwise get creative. I'm sure you can think of something. Use your best judgement.

Remember to say safe, the penalties of failure can be severe. Make sure everyone in your unit is experienced, reliable and trustworthy – one weak link can cause the whole operation to fail. That said, have fun, shadow hide you.



1

by ~

Make every bit count.

Neither high-speed nor high-bandwidth equal high quality, and being bombarded with media does not make us more informed.

Make every bit count.

Why? Because every bit costs energy. A small amount of energy, sure, but those small amounts add up. Once the gas runs out, every last scrap of energy becomes precious.

Make every bit count.

Language is a steady stream of information, and text is the purest form of language. The amount of information you can pack into, for example, 140 characters, is astonishing, the amount of text you can pack into, for example, 140 kilobytes, equally so.

Make every bit count.

Given time, Nature strips the fat from every system. Computer systems are no exception. If it doesn't help an organism to survive in a given environment, it will disappear, become part of the fossil record.

Make every bit count.

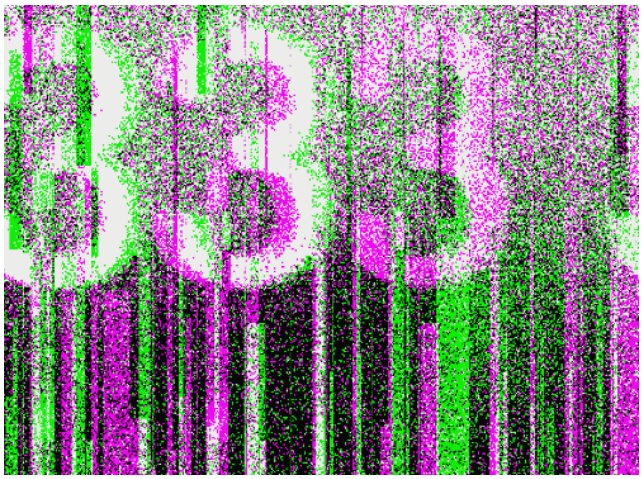
Right now, we stand atop the mountain. We put batteries in our toothbrushes, we use industrial processes to manufacture toys that will be forgotten in a matter of days, added to the mountain. Crunch. But, though the mountain we have built is ugly, the view is incredible. Enjoy it while you can. Take in every detail. Draw a map.

Make every bit count.

A man climbs a hill, not a mountain of refuse but a hill of soil, anchored in place by a thousand trees. He reaches the top, and meets what he has always met on this hilltop: a small stone shelter containing a giant disc, twelve feet in diameter. He takes a key from around his neck and slides it into the hole in the side of the structure. A simple control panel is revealed. One switch, one button. He flips the switch, and the sunlight collected by the solar panel on the roof of the structure and stored by the large battery inside flows into the computer within, breathing life into the delicate silicon organs. He hits the button, and the disc sends out a ping, a single bit of information. I am here.

Make every bit count.

❧

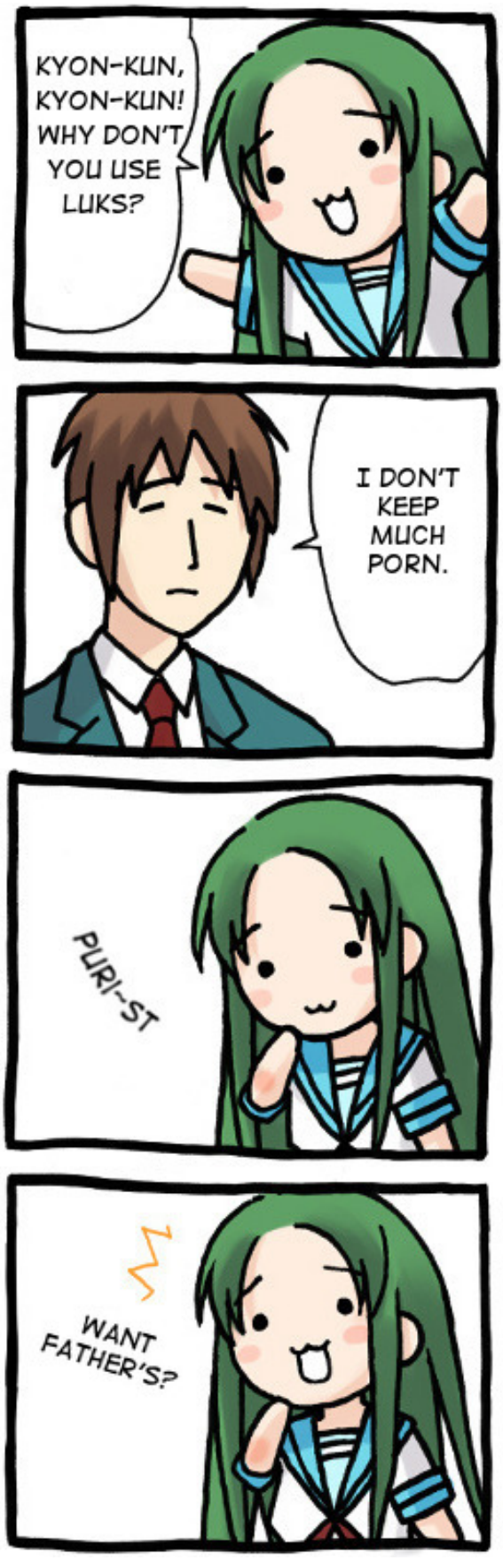


Keeping Application Data Safe with GnuPG

by deterenkelt

Many applications store data on the disk unencrypted or in a way that can be easily cracked. Web browsers, IM, email and WebDAV clients do this, but that's only the beginning of the list. Roughly 80 % of a user's passwords are stored unencrypted. Let's take a look at how Firefox stores them, for example. If a master password is specified, Firefox uses 3DES, or Triple Data Encryption algorithm. However, if no master password is set, the data is about as hard to crack as any plain text document.[11,12] With no master password, files are simply encoded into base 64, which is not meant to encrypt data, but to change its representation to an ASCII string that would be easy to transmit (e.g. embedding small images in HTML files to avoid superfluous server requests).

3DES is considered safe to use until 2020, but it usually requires three different passwords, and 3DES' strength highly depends on the strength of those passwords.[13,14] A short password offers the same security as no password, because not only does specialized cracking software exist for 3DES encrypted files, such software is freely available.[15] This means that if we want our data to be secured, we have to use a long master password for each program that supports a good type of encryption. And what about the programs that don't? Moreover, remembering dozens of long master passwords for browsers, IM, and email, as well typing them every time a program starts? That's a bit inconven-



11 *How browsers store your passwords (and why you shouldn't let them)*. http://raidersec.blogspot.in/2013/06/how-browsers-store-your-passwords-and.html

12 *Hacking / Recovering Firefox Saved Passwords*. http://realinfosec.com/?p=132

13 *Master password encryption in FireFox and Thunderbird*. https://luxsci.com/blog/master-password-encryption-in-firefox-and-thunderbird.html

14 https://en.wikipedia.org/wiki/Triple_DES

15 FireMasterCracker. http://securityxploded.com/firefox-master-password-cracker.php

ient. Fortunately, keychains exist for dealing with this; however, keychains may lack support and still require logging in – certainly not the most convenient. And what's the need in a keychain, if GPG is already there? All that is needed is to encrypt some files holding account information, and decrypt them temporarily somewhere outside the disk. This article covers an example of such a setup.

Figure 1 illustrates the Firefox encryption scheme. It uses two files, `key3.db` and `signons.sqlite` which reside in the profile folder. The former contains hash and salt, while the latter website logins.

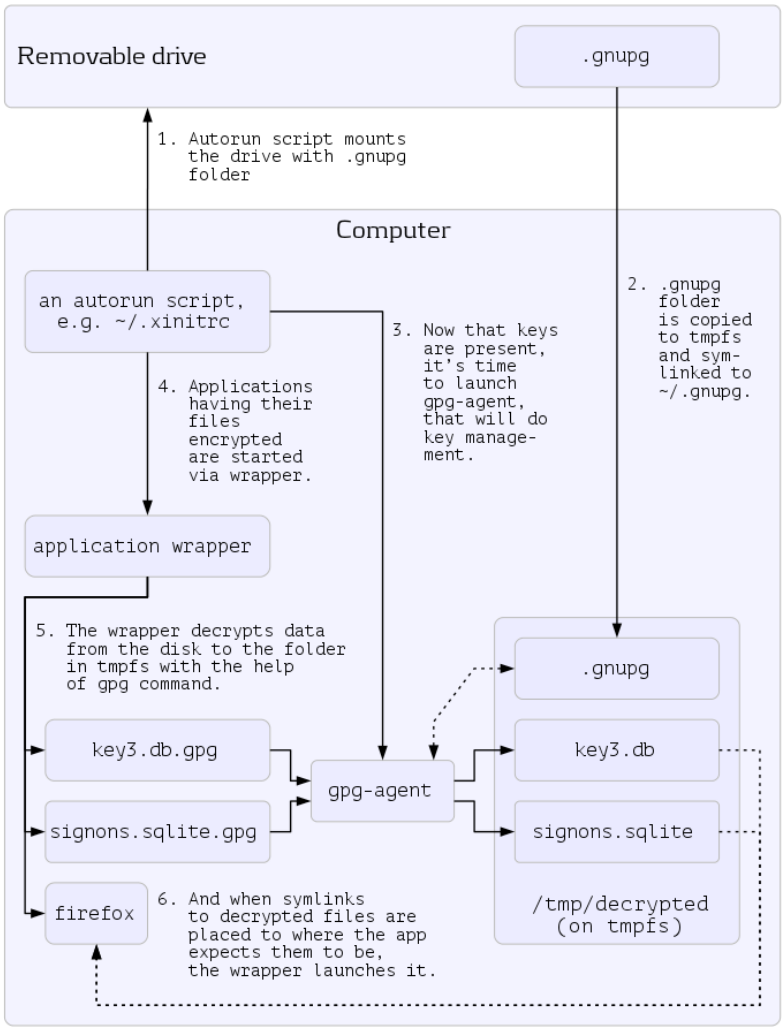


Figure 1

Prerequisites

This article is aimed at people who are experienced with GNU/Linux, do some shell scripting and have already installed and set up GnuPG.[16]

Autorun Part

Upon the autorun script two tasks are laid: ready `gpg-agent`, and modify the environment for the application wrapper script. The wrapper doesn't have to be called from the autorun script. For demonstrative purposes, let's say it's called somewhere at the end of the autorun script or right after it.

There are many ways to set something to run after login in a GNU/Linux OS. Most preferred would be with `~/.xinitrc` or `~/.Xsession`, since they're executed right after X starts, but before it launches any window manager, hence they're best suited for altering the environment. These two files and starting routines are well described in the *X Windows System Administrator's Guide*.

First of all, we should find and mount our removable drive, which may be a flash stick, smartphone SD card, or any other removable media. Let's say that the volume on the stick is labeled '`PHONE_CARD`' and we mount it to a directory named `~/phone_card`.

```
grep -qF "$HOME/phone_card" /proc/mounts && \
    sudo /bin/umount $HOME/phone_card
```

16 wiki.debian.org/Keysigning

We must be sure it is not mounted and there is no 'stuck' record, while the device is not attached physically. `sudo` is important, because most will work from an ordinary user environment (and that's the right way to do it). In order to make `sudo` run this and some further commands automatically without asking for the root password, several lines should be added to `/etc/sudoers`.

```
Cmnd_Alias ALLOW_ME_THESE_COMMANDS = /bin/mount, \
                                     /bin/umount, \
                                     /sbin/findfs
User_Alias ME = your_user_name
Defaults:ME env_reset
Defaults:ME env_keep += SHELL
ME ALL = (root) NOPASSWD: ALLOW_ME_THESE_COMMANDS
```

Don't forget to replace `your_user_name` with an actual name.

```
rm -f $HOME/phone_card
mkdir -m700 $HOME/phone_card
c=0; until grep -q $HOME/phone_card /proc/mounts || {
    disk=$(sudo /sbin/findfs LABEL=PHONE_CARD) && \
        sudo /bin/mount -t vfat -o users,fmask=0111,dmask=0000,rw,\
codepage=850,iocharset=iso8859-1,utf8 \
        $disk $HOME/phone_card
    }; do
    sleep 1 && [ $((c++)) -gt 300 ] && break
done
```

A simple `until` cycle with a counter variable. The cycle body evaluates until `$HOME/phone_card` is found in `/proc/mounts`. The body simply checks for a timeout of five minutes (300 s). Now to the condition. There is a subshell call for `/sbin/findfs` that will attempt to find a volume with label `PHONE_CARD`, and, if it does, the path to the actual device (e.g. `/dev/sdc1`) will be placed to a variable named `disk`. Then this name is used in the `mount` command, which sets options and filesystem type to vfat. The `users` option is to make userspace daemons happy, otherwise it won't be possible to unmount the flash card from Thunar, for example; `fmask` and dmask are usual parameters forbidding file execution on this medium, finally `codepage` and `iocharset` are important for correct name handling on conversion to UTF-8. The list of encodings can be seen in the kernel configuration menu in File Systems —> Native Language Support. The settings above are for Western European languages.

```
rm -rf /tmp/decrypted &>/dev/null
mkdir -m 700 /tmp/decrypted
cp -R ~/phone_card/.gnupg /tmp/decrypted/
chmod -R a-rwx,u=rwX /tmp/decrypted
sudo /bin/umount $HOME/phone_card && rmdir ~/phone_card
```

`/tmp/decrypted` is the directory where `.gnupg` and all decrypted files are to be placed. It resides in `/tmp`, which has to be of the temporary filesystem type, tmpfs. That's the simplest way to keep files in RAM while the computer is running. A symbolic link to `.gnupg` is placed in the `$HOME` directory and file permissions are fixed because vfat doesn't understand UNIX permissions. `chmod` is done recursively on that folder, first depriving all from any rights, then giving 600 to files and 700 to directories for the user running the script. After all the files are copied, the card is removed and the folder is deleted (that last line is optional, the card can be left mounted, so it'd be possible to unmount it when leaving the computer).

Edit the `/tmp` related line in `/etc/fstab` to this:

```
tmpfs /tmp tmpfs defaults 0 0
```

Somewhere around here, you should set up the keyboard layout, because if the passphrase contains symbols that are not present in some default PC101 layout, you won't be able to enter them.

```
export GNUPGHOME=/tmp/decrypted/.gnupg
pgrep -u $UID gpg-agent || eval $(gpg-agent --daemon --use-standard-socket)
export GPG_TTY=`tty`
```

First, `GNUPGHOME` is exported, pointing at the directory where it should look for keys. `pgrep` checks if a daemon is already running – there's no need to restart if it was left over from a previous login, and if it's not present, it's to be launched. Current version of GnuPG (2.1.6) doesn't rely upon `GPG_AGENT_INFO` completely, but it seems to be broken, so the code above refers to 2.0.26-r3 as the most recent stable. Don't forget to put `use-agent` in the `gpg.conf`. The recent versions of GnuPG should also start the agent automatically when `gpg` is called, and they do, but later calls to `gpg` don't seem able to connect to the agent, the example above is guaranteed to work for 2.0.26. The output of `gpg-agent` is `eval`'d because it exports an important variable containing the path to the agent socket. For instance,

```
GPG_AGENT_INFO=/tmp/decrypted/.gnupg/S.gpg-agent:19667:1
```

GnuPG since v2 uses the name `$GNUPGHOME/S.gpg-agent` instead of a temporary directory with a random name in `/tmp`, but in 2.0.26 `gpg` still isn't satisfied unless it finds the address to the socket in `GPG_AGENT_INFO` variable.

This variable shall be present in the environment of every program that wants to decrypt anything with keys that `gpg-agent` holds. At least until GnuPG 2.1 is fixed. However, there is another necessary variable, `GPG_TTY`, that gets exported *to the current environment*, too. It should be stressed, that only those programs that are launched from this shell or its descendants will inherit these variables in their environment. This is why the `~/.xinitrc` or `~/.Xsession` way is preferred: autorun scripts in desktop environments may be run not in the same shell that spawned the DE session, and the environment will get alternated in its own 'branch' that other applications won't know of. In order for `GPG_*` variables to get into every shell on your workspace, they must be exported in the same shell that spawns the DE or earlier. To think of it, this export may take place even in `~/.bashrc`, but it would mean that

the whole thing has to be there, and that'd be a very bad idea.

```
export PATH="$HOME/bin:$PATH"
```

`~/bin` folder will contain the wrapper script. Here the autorun part ends, and it's the last place to set up the keyboard which must be done before the first call to the wrapper, because when pinentry (the program that is used to ask for a passphrase) window pops up… Right, the keyboard will be still in some default Xorg keyboard layout, and if the passphrase is good and contains characters that are not present on it, they will be impossible to enter.

Wrapper

The scheme is as follows: there is a request for a program, let's say, `firefox`. Since `~/bin` is the first directory in `PATH`, it is checked before other directories. The shell will look for a file named '`firefox`' there. And we want to run a wrapper instead of Firefox, so how do we do that? We create a symlink named '`firefox`' that is pointing to the wrapper in the same folder, and in the wrapper we distinguish between applications by the `$0` variable which is the script name. But in this case, it will be the name of the symbolic link the script is called as. How do we avoid recursion while attempting to call actual Firefox, when we're done with decrypting its files? It just needs to be called with the full path, like `/usr/bin/firefox`, or whatever '`which firefox`' says.

In general, to launch several applications in an autostart script, the following code may be employed:

```
startup_apps=(firefox thunar pidgin)
# ...
for app in "${startup_apps[@]}"; do
    pgrep -u $UID -f "^$app\>" >/dev/null || { (nohup $app) & }
done
```

`pgrep` checks whether the application is already running and if it isn't, it's launched via a forked subshell with `nohup` (to prevent it from dying after its parent shell closes). The user id and the regular expression are to distinguish between applications that do not belong to the current user and those whose command line does not coincide with the simple file names as specified in the array above, i.e. to not confuse Bob's `firefox` with Alice's and `mpd` with `mpdscribble`.

Let's look at the wrapper script internals.[17]

`run_app` is the function for running the actual application and decryption procedures. It works for every program. It takes at least two arguments: `$1` is absolute path to the actual binary and `$2..n` are absolute paths to the `.gpg` files with secrets that reside in the same directory as their unencrypted contents.

17 The snippets of code that follow were edited and focus on the key parts of the system. A link to the full listing can be found at the end of this article.

```
local app=$1
[[ "$app" =~ ^.*/.*$ ]] || {
    echo "This function must be given an absolute path \
to an actual binary, like /usr/bin/... or so." >&2
    exit 4
}
shift 1
```

Next, the files are decrypted.

```
for gpgfile in "$@"; do
    [ -e "$gpgfile" ] && {
        gpgfiles[${#gpgfiles[@]}]="$gpgfile"
        tmpfile="/tmp/decrypted/${gpgfile##*/}"
        tmpfile="${tmpfile%.gpg}"
        tmpfiles[${#tmpfiles[@]}]="$tmpfile"
```

Clearing these variables is important if you alternate them, e.g. to add SCIM support. Pinentry has a known bug that causes input to be impossible sometimes.

```
GTK_IM_MODULE= QT_IM_MODULE= gpg -qd --output "$tmpfile" --yes "$gpgfile"
```

Saving last modification time for decrypted files to know whether they were changed afterwards.

```
lastmods[${#lastmods[@]}]=`stat -c %Y "$tmpfile"`
```

...and symbolic links placed to where the application expects plain files to be.

```
        ln -sf "$tmpfile" "${gpgfile%.gpg}"
    }  # end of "test -e"
done # end of "for gpgfile"
```

If the same wrapper hangs in memory, wait for five seconds and then bail out.

```
while pgrep -xf "^bash ${0##*/}$"; do
    sleep 1
    [ $((i++)) -gt 5 ] && break
done
```

Testing if the actual app is still/already running, not allowing a second instance.

```
pgrep -axf "^$app$" || "$app"
```

When the application closes, check whether it has modified the files.

```
for ((i=0; i<${#tmpfiles[@]}; i++)); do
    [ ${lastmods[$i]} -lt "`stat -c %Y "${tmpfiles[i]}"`" ] && \
        modified=t
done
[ -v modified ] && {
    for ((i=0; i<${#gpgfiles[@]}; i++)); do
```

Sign, encrypt to hidden recipient, and place the updated file where the old one was.

```
GTK_IM_MODULE= QT_IM_MODULE= gpg --batch -se --yes \
    --output ${tmpfiles[i]}.gpg \
    -R *$USER ${tmpfiles[i]} &>>$elog || \
    echo "GPG couldn't encrypt ${tmpfiles[i]##*/}. See $elog \
for details." >&2
```

Not `mv`, because it erases symlinks.

```
        cp "${tmpfiles[i]}.gpg" "${gpgfiles[i]}"
done } # end of check for modified files
```

Erasing files from tmpfs. This may be done here, or the whole `/tmp/decrypted` can be erased on logout (if the computer is not to be switched off).

```
for ((i=0; i<${#gpgfiles[@]}; i++)); do
    rm ${tmpfiles[i]} ${tmpfiles[i]}.gpg
done
# ... end of run_app()
```

And here are the actual first lines to be executed when the wrapper starts. They get the name of the app the wrapper should start as and enable logging.

```
app_name=${0##*/}
elog="/tmp/runapp_$app_name"
echo >$elog
exec &>$elog
set -x
```

As simple as that. Each application is a `case` statement, which decides the paths to an actual executable and encrypted files for a program.

```
case $app_name in
    # NB: only actual binaries with absolute paths here!
    firefox)
        [ -e /usr/bin/firefox-bin ] && \
            firefox=/usr/bin/firefox-bin || firefox=/usr/bin/firefox
```

`$@` are the arguments passed to the command. When some other application runs firefox with arguments, it usually wants to open a link, so if Firefox is already running, don't start another wrapper and send a command opening the link in a new tab.

```
if pgrep -u $UID -xf "^$firefox$" &>/dev/null; then
    $firefox -new-tab "$@"
else
    run_app $firefox \
        ~/.mozilla/firefox/profile.xqwzts/key3.db.gpg \
        ~/.mozilla/firefox/profile.xqwzts/signons.sqlite.gpg
fi
```

Don't forget to replace `profile.xqwzts` with an actual profile folder.

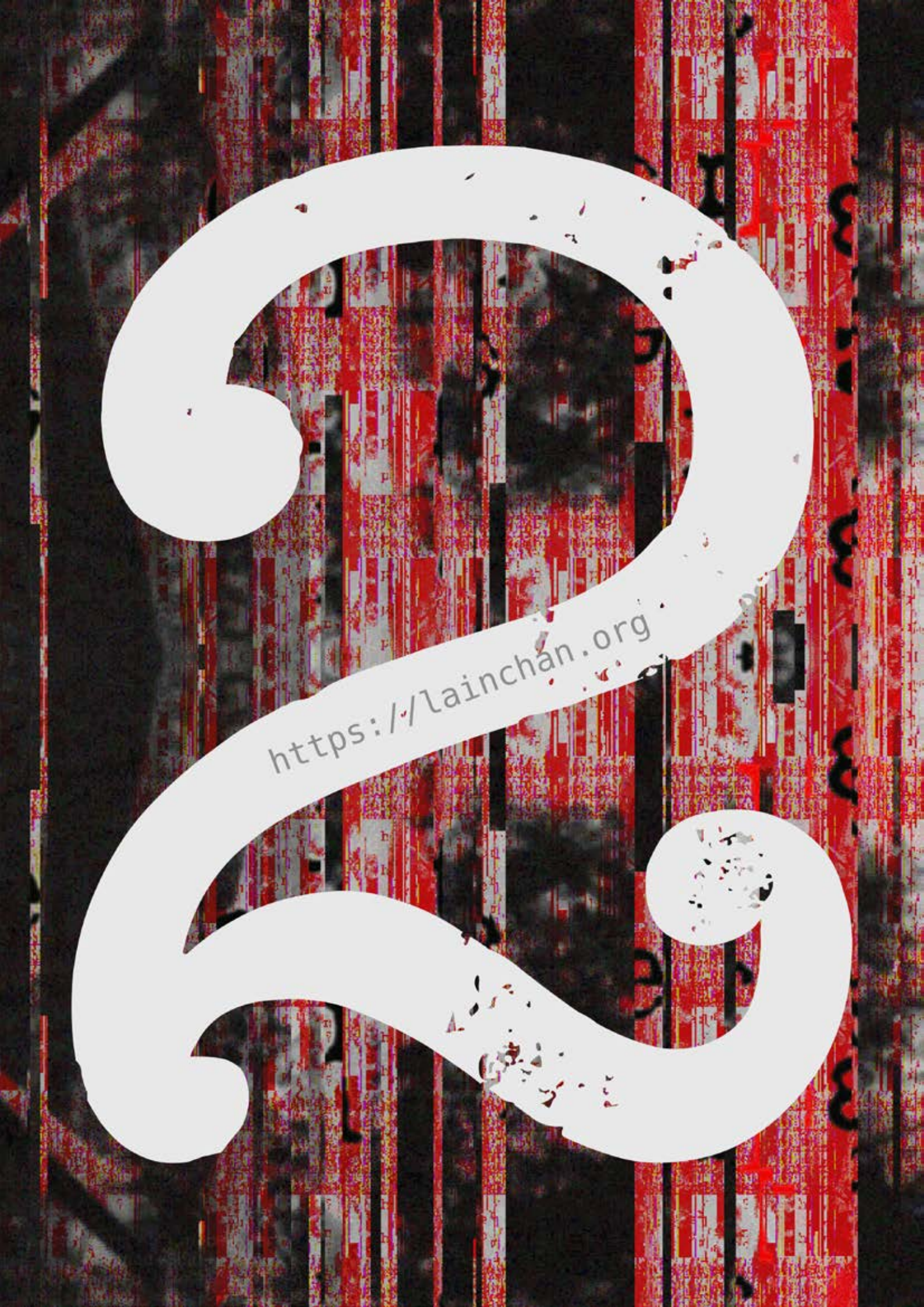
```
    *)
        cat <<"EOF"
Usage:
    Just use symbolic links:
    ln -s ~/bin/run_app.sh ~/bin/firefox

Don't forget to `export PATH="$HOME/bin:$PATH"`!
EOF
        exit 3 ;;
esac # $app_name
```

Addendum

This setup isn't meant to form a bastion of safety, it's aimed at a compromise between convenience and securing files on workstations other people may have access to. The other purpose is to make it safer to backup on cloud services. This setup is vulnerable to attacks from the code users run by their own will, e.g. if a malicious script wants to run something with GPG, it will decrypt files automatically. Of course, the script must know that the files are encrypted with GPG and where they reside. Or that `~/bin/run_app.sh` is being used. If you are truly afraid that somebody has access to your computer when you don't, use keychains and make them ask for a passphrase each time such a request comes. Or better don't use them at all and perform all encryption operations manually.

Complete File Listings

- https://github.com/deterenkelt/dotfiles/blob/3ae623d/.preload.sh
- https://github.com/deterenkelt/dotfiles/blob/f035f3d/bin/run_app.sh


https://lainchan.org