end the distro war, end the distro

	shout out to mgk fans	(yes,its basically irrelevant)
so this isnt about killing your distro. this is about evolution.	your distro is free software, so as long as people want to work on it, people will. unless someone stops them. **==>**	this isnt about how to do that, but how to prevent it. in this sense, killing the distro (or just letting it die) is more like killing yoda or obi wan.
before we get started, a few givens and prefaces:	i am a free software advocate, gone rogue. i dont toe the party line. but i do care about freedom.	this is not about open sores, "loving linux" or cozying up to monopolies. this is mostly about gnu.
i want everybody to learn how to code. i want to make that easier. ive spent years working on that.	i worked on this more while i waited for init freedom to reestablish itself after systemd took over debian.	before that, i was distributing debian by giving away used laptops and desktops.
silicon valley always wants everyone to learn how to code.	they want to continue to hijack education as a tool to further monopolies and reduce salaries.	teachers are fighting this. i want humanities to include technology, not for schools to breed c# monkeys.
philosophy needs to include technology. it is too relevant to treat understanding computers fundamentally as a job skill.	computing isnt just a job skill any more than music or art are just job skills. computing is ubiquitous in modern life.	if schools dont teach people how to understand computing, corporations will have a monopoly on that. this is bad news.
but thats about me, and where im coming from. this is about the future of the distro, and free software.	but if you teach everyone how to code, it will be a lot harder for monopolies to exploit their freedom.	monopolies deal in cheap abstractions to"help"people who feel helpless. coding gives them more power.
with coding, you can make your own cheap abstractions!	lets talk about another given, copyleft. we all know that monopolies find it less attractive. that makes it useful to software freedom. i am not anti-copyleft like the bsd community.	in some contexts i promote permissive licensing... for reasons im comfortable with. no arguments i will make here depend on this-- use copyleft as often as you think it is important!

the four freedoms are vital-- freedom to **use**, **study**, **change** and **share** shouldnt be compromised.	everything said here tries to bow to those freedoms. use the official fsd if you like. we will call it "remix."	the goal isnt to change the fsd or ignore the 4 freedoms. they are worth repeating. but we will use "remix" to refer to these.
any time this document uses the word "remix"-- semantics aside, we are referring to all 4 freedoms.	the 4 freedoms are a vital baseline for freedom. we are less free, if one freedom is compromised.	open source turns the 4 freedoms into "open" and then piggybacks other things onto it.
you can have "4 freedoms" but its not "really open" unless theres a way for people to collaborate-- wait	4 freedoms just means that you are free to do what you want with the software. it doesnt promise support.	support is a nice extra! its ideal. but it has much less to with whether the software is really free.
stallman says that documentation should be free-- as a free culture advocate, i strongly agree. not because "the software isnt really free otherwise."	be extremely careful what you piggyback onto the 4 freedoms. it can be used to put other priorities over actual software freedom.	for example, say im a communist. or a furry. and if youre a communist or a furry, you might think thats great. its irrelevant to software freedom!
software freedom is established when you have (and guard) the 4 freedoms.	you have to be careful how you guard them. if i piggyback "rights for furries" onto the fsd, now...	1. free to use 2. free to study 3. free to change 4. free to share 5. free to be a furry
i have nothing against furries whatsoever. but how well my software caters to furries has sweet f-a to do with the 4 freedoms.	but heres where it becomes a problem-- say there are two projects. one cares about your freedom, and the other open source bs.	weve changed the definition of software freedom to make furry freedom a strict requirement.
so now, we can divide communities and projects over the level of furry freedom they offer. that might appeal to furries.	but in the long run, it might also mean that the 4 freedoms become lower priority. that would be unfortunate.	even if you dont agree with me on this, thats ok. i really want to warn people about piggybacking on irrelevant issues though.
i also feel its important to prefact with that, since there are other issues that i consider very relevant to software freedom.	but i have no desire to change the 4 freedoms. they are a good baseline, we should be very careful how we overload them.	if and when people critique this, theyre going to say i want to overload the 4 freedoms. but i think they should come first.
if we do piggyback (and it is already being done) then we should be careful not to sabotage free software with peripheral ideals.	but im not picking on furries either. i am happy for them to contribute to free software and create furry free software sites.	lets be careful not to adopt the open source silliness that makes collaboration freedom #5. free software can be solo. collaboration is a peripheral issue, at times.

for comparison, i promote free culture and free software. i think these are both important. you can promote both at once. i think the two issues complement each other.

heres the thing though-- you have to be a free culture advocate or copyright reformer to care about that. free software is free software, no matter what you think about that.

so go ahead with your piggybacking of issues. just make certain that it doesnt erode software freedom in the process. people will know if that happens. and others will dismiss them.

i think its deeply unfortunate that most free culture advocates seem to use a mac. i think its deeply unfortunate that most fsf fans promote noderivs for an arbitrary irrelevant reason.

but i wont change the fsd over it. the arguments that open source makes that begin "its not really 'open' unless..." are just nonsense. be careful not to fall for that kind of thing. it shifts priorities away.

where this becomes kind of serious is, what happens when you find a way to hurt software freedom while technically meeting fsd requirements? the fsd is a vital baseline, but is it eee-proof, for example?

is it possible to use eee tactics against free software-- even when the software is gpl licensed?

first, i believe it is. also, i will talk about why i think it is. but above all, this document is about how to defend all free software against eee attacks. thats why we are doing this.

i do think that systemd is an eee attack on free software. i think a lot of the damage is already done-- we have lost years of progress over this eee attack.

on the whole, i feel optimistic about moving past a chapter in software history where systemd is foisted on us. the whole idea that there was some kind of vote and we lost sidetracks the issue.

many of us dont want to use systemd. no one has to cater to that per se, but for many of us (and i think the number is growing) this is a software freedom issue. eee is a software freedom issue, and i think it is eee.

but the point of this is to fight eee attacks, not prove that systemd is one. even if we did prove that systemd is bad for software freedom, there will be other attacks like it. monopolies dont just stop attacking us.

by all means, we are not done fighting systemd, but lets learn from it and bolster our defenses against eee while we are at it. lets have more freedom.

so im going to talk about things you may not think are really a problem. and im going to talk about how to work against (solve) those problems.

you might think these are non-problems not worth solving. thats ok too. i think it will increase freedom and bolster defenses against eee, but perhaps im wrong.

the gpl is code. it is legal code, but it is code. and while a lot of code is compiled and tested, legal code is interpreted and tested-- by people, in life.

computer code doesnt always yield the expected results, but it works based on logic. legal code also doesnt always yield expected results.

like most or all code, legal code has vulnerabilities that can be exploited. tivoisation was an exploit of a vulnerability in gpl2. the gpl3 patched against this.

eee is a class of exploits used to attack a monopolys competitor. free software competes with monopolies.

free software is resistant to eee attacks. copyleft increases that resistance. some eee attacks do work.

in practical terms, strong resistance to eee means that fewer eee attacks will work. years may go by.

lets talk about marketshare. open source thinks its important. we do too-- we wouldnt call it "marketshare" though.	to us, marketshare means "all software should be free." that applies to the ecosystem-- the marketshare of freedom itself. we dont call it that.	for a monopoly, marketshare is what you get by creating lock-in and dealing unfairly with competition. for open source, you get it by emulating non-free heroes.
we just want all software to be free. so "marketshare" for one "product" isnt relevant. but something similar does matter for the entire "faif" ecosystem.	thus, we dont technically care how many users any particular individual project has, as long as its enough to sustain development and maintenance.	but we do want everyone to make use of the free software ecosystem instead of non-free software. a freedom-respecting plan to make that happen is good.
most plans to make that happen either dont work, or dont really respect your freedom. its not a small goal.	feature creep is related to eee. unix was chosen as the operating system to make gnu from in part because it is very portable.	unix was designed to be ported to as many things as possible. this meant very small, simple programs.
feature creep would have defeated making unix work across so many systems. it would have lent itself less to the gnu project also.	the connection there could be disputed or less than obvious, but there is a more recent example: the mozilla web browsers.	mozilla produces free software. it also produces feature creep. i use mozilla products, and i hate them very passionately.
i didnt always hate mozilla though. i have watched them turn into something i consider hateful. nonetheless, we are talking about free software (or close enough to fork it.)	chrome has popularity (it is non-free, and i also think it is very untrustworthy software-- i dont even trust chromium. i dont trust google.) but for whats free, mozilla is prominent.	perhaps mozilla thinks that they need feature creep to "keep up" with other browsers. they also introduced eme, which hurts freedom to keep up. but thats not all.
feature creep produces some lock-in with the mozilla family of browsers, even if you fork it.	feature creep still makes mozilla harder to fork, harder to maintain, and harder to switch browsers.	this brings us back to the 4 freedoms and piggybacking. do i think freedom #5 should be freedom from feature creep? no!
the 4 freedoms are a clear baseline-- and "no feature creep" would make the entire fsd ambiguous.	of course you can make free software that has feature creep. but be aware-- it eventually hurts software freedom.	the unix design philosophy was not developed for the sake of freedom but it does eschew feature creep. that helps preserve freedom.
so i dont want to change the fsd, but an argument for freedom is being made against feature creep.	i use mozilla for being free-- it runs js per-site, lets me control image loading, tabs. and private mode.	very few free browsers run js per-site and control image loading and have tabs, and private mode.

so by the time i want just those features, im pretty much using mozilla. there are a few other browsers with similar features.	most free browsers with that list of features are 64bit or use weird controls. i used vimperator, i dont want anything weirder.	but i think plugins made mozilla an addictive option to people. and now pdf.js, the bloated mess that is html5-- web browsing produces lock-in, i believe.
certainly we can get past this, and it think making browsers based on python and webkit will be key.	though i use a browser i hate more every year, and switched to pale moon up until the lies about noscript-- worse examples of feature creep exist.	we can use feature creep to hurt software freedom deliberately, and turn good projects into ones that suck like mozilla. we can also go in the other direction, like gnome 3.
im not going to take time to properly address gnome 2/gnome 3/mate here, gnome was already bloated and hard to fork back when i actually liked it.	i liked gnome 2 and i applaud mate for trying to preserve it. i think its rotten what debian did to the mate devs-- it was petty, and damaged our choices.	but it has to be said that gnome has long been an example of the sort of problem im talking about. is that ok? well, "its ok until it isnt." thats feature creep.
gnome is a project that is extremely arrogant and damaging, but theres just no good reason to get very far into it here.	**<==** even without the previous comment, i know im going to deal with fans of gnome saying the rest of what i said is baseless.	the truth is, ive talked about gnome many times, sometimes in better detail, but i dont have much to tie in about gnome right now.
after all, this is about distros. now it will be possible to misinterpret, exaggerate or take what i say about distros out of context.	technically speaking, a distro is a harmless thing in and of itself, i sort of maintain a few. we can get into all kinds of nonsense about terminology.	the first "distro" i ran was tomsrtbt, and for reasons not directly related to this, i think the boot floppy is the best way to examine this concept of the distro.
because people conflate the concept of the distro itself with the modern implementation/example.	i think it is far more useful and illustrates more to consider that a distro is nothing more than a suite.	or a distro is nothing more than a collection of software. any other definition projects extra things.
i personally tend to think of a distro as a bootable collection of software. thats very convenient and very conventional.	so if i take debian and produce a version that requires you to first install a bootloader, is it no longer a distro then?	perhaps it is more useful to think of a distro as something inherently bootable, but lets consider accuracy even if we do so.
if you ask a distro maintainer (or fan) what a distro is, they might argue that a distro has its own repos too.	for creating a list of repos, this might prevent too many fly-by-night niche efforts. gnuinos is the only devuan version i know...	...with its own repos. the website is outdated and i dont know if the repos are. all the other derivatives are not distros then.

and then devuan isnt a distro either, because it uses debians repos. but it supplements them. ok, so if you use debians repos but add ten packages like refracta, you are a distro.	actually, that debate will probably never be settled. so the only truly definitive concept of a distro is probably "a collection of software." well, crap.	either way, i think there are practical advantages to considering that any boot disk with user programs is a distribution. this prevents a lot of pretentious non-requirements.
of course you can have classes of distro, like ships on star trek. debian is clearly a galaxy class distro. puppy is at least more than a shuttle.	lighterweight distros have a reputation for working better on older hardware. as someone who refurbishes hardware, i can find merit in that.	i also think lighterweight distros are easier to remix, though before i created my own tools for remixing distros, i refurbished old machines with debian, lxde.
there is probably a diminishing law of returns on what is lightweight. the oldest machines get harder to find and slightly newer ones get more common.	i rarely find a piii and for a few years i wanted to play with a 486 again (at least once) but i found a couple piiis and a pii but zero 486s. sure, theres ebay.	but for example, i think both debian and puppy are less trouble to remix than tinycore. if lighter always meant easier to remix, tinycore would be easiest.
puppy has a reputation for being the fastest-- imo, only antix (not puppy) reminds me of the ridiculous speed i knew puppy for in 2006.	antix is ridiculously fast, but i have no desire to remix it (i did remix it, once. for about a day.) id like to know how they made it faster than puppy.	it seems to me the kind of knowledge used to make antix so fast, could help all distros if they would document it. i could take it apart, someone should.
but i can mix distros together, which is one reason to use the word "remix" instead of remaster. theyre really the same thing, but the word "remix" fits in these boxes.	feature creep and lock-in are two reasons to let the distro concept die, if something better comes along. i think something will come along, i have thoughts about what.	people can always make distros if they want to, and i think some of the things that come along will improve the distro. if they improve it "too much" (actually, its fine) it would be less distro-like.
but if we go with the definition of "bootable software collection," i certainly think we will still have that.	i think we will still have and make our own bootdisks. i do think it could be a lot easier.	people who are used to remastering may think its easy enough already like it is. i dont, not at all.
all i had to do years ago to make a bootdisk was copy the floppy and then put programs on it.	you can definitely do that with something like isomaster. lets look at that option, i was once a fan.	i actually used isomaster the other day, as a tool for troubleshooting the sfs portion of a live dvd.
when i first used puppy more than a decade ago, isomaster was very useful.	my issue with isomaster is that you cant script it. it requires a gui environment.	i have fixed both tof hese things. its fun that it was still useful this week.

mostly i dont use isomaster, i havent used it much in years. i bet a lot of people still love it.	what i wanted is to be able to open a dvd, copy files to it, and close it. i dont mean with the mouse. i mean like a filesystem-- a scriptable version of isomaster.	dont worry, i have this capability and have used it on a variety of distros from puppy to refracta to trisquel to tinycore and void linux. it works.
one thing isomaster wont do is open the sfs files. so you open the iso, you open the sfs files, theres a little more to it than that just so you know-- thats the idea.	i have no complaints that these steps are necessary. i manage them just fine. for years, i didnt want to create my own distro because this is too tedious.	there are other more official ways to remaster too. theyre usually more technical and complicated. either way, i wasnt up to any major remastering.
automating the process is the only way i became interested. i didnt set out to remaster anything, really. i was working on a demo program for my programming language.	with all the many versions of puppy linux out there, i wanted to create a program that would open them, look for certain features and report on many versions in a large html table.	this grew into a program that could remix puppy, and even mix it with refracta. so i wanted to know-- could i update puppy using refracta binaries?
to some extent, yes. and i know (i have preached) all the reasons to not do this. i know about the problems in mixing the binaries from different distros or even versions of the same distro.	yet, people do. and while it is not a requirement for this casual thesis, it is a feature of it. i know of absolutely no other software community so dedicated to remix as the one for puppy.	thus, while i dont use puppy these days, i still remix it. i demonstrate my automated remastering, i try to lend a hand occasionally.
puppy has a much more powerful tool called woof. it can do everything-- thats awesome. unfortunately, it requires you to do everything. thats less awesome. few use it.	again, this is not a complaint or request for people to fix woof to suit my purposes. i simply think there is room for improvement in the state of the art.	and i think there will be improvement, so far from a feature request, i am trying to peek at the future. i have experience with this with regards to computing.
from a technical standpoint, we can make it easier to open and edit every distro. but either way, i have tools that get closer to doing that.	just automating opening the iso, opening sfs files, copying to them and closing the sfs and iso is a big deal.	for puppy i avoided chroot, because i was less familiar with it and didnt know if it was fully scriptable either (it is) and im not sure chroot from puppy let me change apt packages.
without getting into some debate about kernel and binary versions, let me just say that originally i toyed with mixing puppy tahr and refracta based on debian 8.	i now also use chroot for remixing trisquel-- the remix goes a little farther than previous examples, actually changing the init. but mostly ive avoided it.	to deal with apt packages without chroot, from puppy, i used dpkg-deb to open the packages and then include their contents in the sfs.

these details are barely relevant here, though i thought id mention them for completeness.	heres what happens though-- when i want to remix a distro, i take a script of about 1000 lines.	then i tweak it for that distro or distro family, and the closer it is related the less tweaking it requires.
i could spend from a day to a week (usually less) adapting the script to a new distro.	then i have a new derivative. i can add to it, i can change it, i can remove it.	compare this to refracta-- refracta lets you change the system and then create a snapshot.
but you have to boot the distro youre remixing first. then you get a full iso.	i should point out, not just to be nice but because its true, that im mentioning refracta because its a great snapshot tool, my favourite installer ever, and the maintainer is a great guy.	if you want a snapshot tool, you couldnt do much better in my opinion. if you want an installer, i dont prefer anything else. but i dont want a snapshot tool. its just not what fits my usage.
again, you might think the disadvantages of remaster tools or snapshot tools are so minor or irrelevant, they arent disadvantages. still, i think we could do better.	the snapshot tool produces an iso file. so does automating the remix. the difference is, the automated remix lets you choose:	distribute the iso, or the script that creates it, or both. the script will be about 100k. the iso will be at least 100 to 2,000 to 6,000 times that, or more.
heres the first taste of extra freedom though-- the automated remix gives the user way more control over what it does.	suppose a few lines of code install icewm and change its settings. if im fancy about it, i can put all that in a conditional / option.	then all you have to do is remove or change the part that turns the option on. even without that, you only have to remove those lines.
since every modification is scripted, changing the script changes the distro. unlike a normal remaster, every change is documented, reversible.	it allows you to make important tweaks to a distro without bothering the maintainers or getting serious bandwidth for hosting your version.	i have used it to add packages, remove bloat, increase the friendliness of a distro, change the window manager, and overload "help" in bash.
though rebranding was not a priority, i asked the community and they thought a rebranding was good, so i used it for that.	most recently, i took the capabilities a bit further and replaced systemd as the trisquel init with upstart. i could do more.	this helps standardise your environment across multiple distros, without freedom-limiting efforts like systemd.
i also did everything in the easiest way possible. if id done this in bash, it would require more debugging. ive written a bash script half as long-- its tedious.	after more than a decade of experience with bash, i have yet to make it do everything i need with relative ease. its good for what it does well-- a lot.	python is more flexible, but this wouldnt have made it this far as a python script either-- not with me writing it. i wasnt very interested in creating a remaster script!
it grew out of other code.	can remaster apps help?	sure. a new breed of them.

this is the part that many people will think is silly-- or theyll think its redundant, because of things like debootstrap. debootstrap as far as i know is debian-specific-- this isnt-- and harder to use. not aimed at beginners.	of course i also think everyone should learn how to code. a lot of people act or talk like that means everyone should learn how to become a really good coder. what i mean is that people should gain a basic understanding and comfort.	so there is no reason to reject a simple level of coding, or a simple amount of command line. if everyone appreciated why code is powerful, that wouldnt be so uphill. also, people who wanted power would already have it.
but when i talk about understanding code, i mean the very basics-- variables, input, output, very basic math, loops, conditionals and functions.	python is a good first language, though i think python 2 is friendlier. i also knew a former nokia developer who preferred it for his own reasons.	fig is designed around things that slowed down teaching python to beginners-- an experiment to create the friendliest beginner language.
i dont have a monopoly on that. i teach beginners how to create their own languages so they can understand code-- and so they can help me understand what "easy" is.	one thing i think helps is a smaller number of commands to learn before youve mastered the language. fig has < 100. it could easily have > 200 but i put my foot down about it.	of course like the language that inspired it, fig has a shell command, so you can extend it with shell code. you can extend it with inline python. i am light on using the latter.
but this is my best tool for teaching coding. i use it for most of my own coding now. i use it to remix distros as well. that wouldnt have happened with a different language.	i didnt want to do traditional remastering. too much tedious work that is only good for that instance-- unless i use snapshots and save the filesystems in progress.	if you screw up something modifying an installation for a snapshot, you have to figure out how to reverse that-- or start over. you could back it up... but i wanted something that let me make more mistakes. reversing them is changing or removing a line of code.
but in broader terms, this is about automating more of the distro-building/rebuilding process. right now, thats mostly a hardcore effort.	i watched a video years ago where someone at a debian talk suggested that debian use more of the techniques that ubuntu used to streamline building.	of course i understand the difference between building a distro and tweaking a script to maintain a remix. i believe this difference will realistically get smaller.
i describe this as going from building distros to building distro factories-- and from distros factories to "distro 3d printers."	by 3d printers i mean applications that are smaller and simpler than the distros they output. my own script is a simple one.	like a netinst that produces a new live iso instead of installing. is this redundant? i get why someone would say so.
in every instance, some of the work done by maintainers can be in apps.	i am well aware that applications also have to be maintained.	theyre also easier to fork than distributions typically are.

im also aware that automating package building is intense work. so even if everything else became trivial, building the actual repo is less trivial.	but, all of these things move the work involved of distro building and maintenance from "the distro" to applications. this makes distros "portable."	this makes distros easier to fork as well. and it gives the user more freedom. to me, this is vital. but i want to include the advantages for maintainers.
between the advantages for user freedom and reduced effort for maintainers that i think is inherent in all this, its hard to imagine the distro not moving more in this direction.	hybrids are increasingly common, automation is increasingly common, and there is great potential in remastering. i didnt invent this of course-- olpc uses an automated remaster to turn ubuntu into their platform.	so no ones going to be forced to work on distros in a new way. some distros will do more things like this, some will do fewer. but the ones that do more will have less work and more freedom, at least i think so.
i expect sceptical takes on this, and if these ideas take hold i expect debate about the merits of them. any time the user is given more freedom, there is concern it will be misused.	for example, universal packages allow any developer to create a package that will work on any system. theres no way to prevent this-- anyone can create an application to make that work on your distro.	but the free software community seems concerned about this, because they lose the opportunity to create only-free repos. well i get that-- but only-free repos never stopped people from installing non-free software.
a distro only has so much power to keep a person fully free. linux-libre breaks most non-free module installations, but thats considered a bug by its devs.	a fully-free repo doesnt offer non-free software. i prefer fully-free repos. i dont want non-free mixed in. but a repo only controls a repo, not the entire ecosystem.	you can still have fully-free repos of universal packages. now instead of choosing a non-free universal package, people can choose to use yours-- like an fdroid for pcs.
then they might decide they want to use your fully free distro, which is ideal. or they can start there, which is also ideal.	the reality is that we arent presenting or preventing any non-free choices either way. with trisquel i can go to the mozilla website and download a tar.bz.	then i can unzip it into /opt in trisquel. voila-- universal package! i mean, the freedom implications are the same.
one important thing that someone pointed out is that someone can install snap and then say on the command line: snap spotify and non-free software will install from a program in "main." that i agree is a problem.	as a hard rule, programs that install non-free programs should never be in main. thats what contrib and non-free are for. but this isnt perfect, even in trisquel. wget and iceweasel can be used to obtain non-free software.	however, i dont mean to be facetious-- while wget and iceweasel can download non-free software, snap creates a namespace/listing where just typing spotify gives you a non-free repo in main. thats its purpose.
even if universal is ok ==>	snap is probably not.	this debate will continue.

but i do note that this is another direction where users have more freedom over maintainers. when it comes to something like removing systemd, i would love a universal init package!	im not sure a universal init package is going to happen. that doesnt mean it couldnt. for a non-rolling release distro, you could at least in theory take everything people needed to change inits and put it in a package.	this could be done as installer, a universal package, a tar.bz like mozilla uses, or like ive done with mkfreemos 0.1, it can be an automated remix.
even icecat can be downloaded as a tar.bz using wget and unzipped to /opt in a non-free distro. so already this sort of proto-universal-package works in the other direction too.	again, im not defending snap as it is implemented. but the idea of universal packages works with or without the free software community, and (with or without the free software community) it can be used for freedom.	and if the community fails to recognise a threat like systemd, if trisquel spends a year or more foisting something like systemd on its users-- then if its not all up to maintainers, universal packages can reduce necessary forks.
freedom can thus be defended by users and not just maintainers.	most efforts to do so have failed because there is no substitute for caring about the issue of freedom.	if users dont care about freedom, they wont choose free packages (regardless of architecture) and they wont choose trisquel.
i also believe that understanding coding will give people a better appreciation of what they lose when they use something non-free. if this is not true always, it is true more often.	ultimately trisquel makes freedom easier to have-- however, it does so in its capacity as a distro. i think switching to systemd shows that the distros capacity for preserving freedom has limits.	we can surpass those limits outside of the distro concept by producing more applications that allow users to become more maintainer-like. thats what will "kill" the distro. but it probably wont disappear.
most likely, the distro will become a less common concept, less of a given, maybe even cease to be the most popular concept related to the purpose it serves.	but i maintain this would ultimately be a good thing. when i look at the various distros out there, which differ because of various reasons: legitimate freedom/choice/pride ==>	i think of each distro as a pair of brackets: [ ] to some degree, those brackets can preserve and convey culture and goals. eventually, i feel those brackets limit our freedom.
the more it is a priority for a distro to provide freedom-- the more that is an inherent goal of a distro, the longer it will take for the limits created by those brackets to become significant.	but i think there are already signs that those brackets are starting to fail the users. i think even for users who consider freedom vital, the distro concept is not scaling to the diversity of users.	when a distro fails to provide what users need, they migrate. and i think a lot of migration has increased lately. not just among newer distro-hoppers. i have migrated since systemd hurt debian.

i am not against migration per se, but i never thought i would need to migrate from debian. i believe the distro war-- not over differences between distros, but over limitations of distros to provide users what they need-- especially regarding systemd-like problems =>	has resulted in an ongoing and growing free distro diaspora. by no means are all migrations negative. if a choice is available one place but not another, you migrate to where the choices you want are found. thats better than no choice.	but the more it happens, the more i think it indicates that choices and freedom are decreasing. this is very hard to prove-- harder with free software-- but again, i never thought id ever need a distro other than debian. i always thought debian (if only debian) would scale.
systemd made debian smaller in what it offered for freedom. people left. people always leave, but i think it drove away people who never thought it would be necessary to leave.	debian-specific, systemd-specific problems aside, i think every time a distro fails to provide for a use case, migration happens when it reaches significant proportions.	now suppose you take those brackets away: [software] software what you have left is still a distro, without the limitations. sorry for being vague and conceptual here.
i have already talked about what sort of software i think will make this possible. it could be other factors and other designs that actually do it. i think it will be automation and applications that do it.	it could be universal packages that do it. im not married to them yet. i can just as easily imagine a distro that uses 2000 mounted iso files as packages. but tinycores sfs + symlink is more likely.	whether these packages are sandboxed or symlinked or compressed or in iso files (im not very serious about the iso files, its just a wild hypothetical that isnt very practical but could work anyway) isnt the point im making.
while a distro is essentially, fundamentally just a collection of software (typically bootable and/or installable) and a package is more like a smaller unit of distro, speaking as broadly as possible, we make a great deal of specifics.	these specifics form a small culture around each distro. when you choose a distro, you are getting tied (in part, to one degree or another) to that culture. i dont mean in the broader sense of freedom-- i mean in the narrower sense of one group of maintainers.	personally i think we have reached a point where we have handed over too much user autonomy to maintainers-- even well-intentioned ones most of the time. but well intentioned doenst mean capable of enough of suiting our needs or goals.
hence the migration, hence the forks. and while these are typical reactions to the limitations of the distro-- things that almost prove that freedom is working-- because look, you can fork. i dont think thats working well enough either anymore.	when something works pretty well, but starts to show its limitations-- if and when we reach "peak distro" (did someone throw a tomato? alright...) people will complain more and more than distros arent good enough. this isnt just a lack of gratitude.	i think its a realisation or intuition (in technical terms, not to imply that users are psychic) that we can somehow do better than this. not just by trying to improve the distro, but with new ideas. freedom includes experimentation. good luck-- happy coding.