چرا پیتون ؟

کاردینال بیگلز ، اریکِ بی ایمان را بیش از چهار ساعت روی صندلی راحتی نشاند تا سرانجام این اعترافات را از او گرفت ...
اولین نگاه من به پیتون یک تصادف بود ، و علاقه چندانی به چیزی که در آن زمان دیدم پیدا نکردم.اوایل سال 1997 بود و کتاب " برنامه نویسی پیتون " نوشته مارک لوتز از انتشارات O'reilly به تازگی بیرون آمده بود.کتابهای O'reilly به ندرت به در خانه من میرسند ، که در آن مورد هم توسط یک فرد ذینفع درون سازمان طی یک فرآیند تصادفی که من تصمیم گرفته ام دیگر سعی در فهمیدن آن نکنم از بین تازه های چاپ برای من فرستاده میشوند .
یکی از این کتابها " برنامه نویسی پیتون " بود. از آنجا که من زبانهای رایانه جمع آوری می کنم این موضوع برایم جالب بود. من بیش از دوجین زبان همه منظوره بلدم ، برای تفریح مفسر و مترجم های زیادی نوشته ام و شخصا تعدادی زبانهای یک مختلف طراحی کرده ام. تازه ترین (Markup) منظوره و فرمهای نشانه گذاری پروژه ای که من در زمان نوشتن این مقاله به پایان رسانده ام ، یک زبان یک PNG (Portable Network برای کار کردن روی تصاویر SNG منظوره به نام را در آدرس SNG است. خوانندگان علاقمند میتوانند صفحات خانگی (Graphics مشاهده کنند. من همچنین چند نسخه پیاده http://www.catb.org/~esr/sng/ Retrocomputing سازی شده از چند زبان همه منظوره عجیب را در صفحه خود به آدرس Museum http://www.catb.org/retro/ قرار داده ام.
من قبلا آنقدر درباره پیتون شنیده بودم که بدانم از آن چیزهایی است که امروزه به (built-in) آنها زبانهای اسکریپت میگویند، یک زبان تفسیری با مدیریت داخلی حافظه و تسهیلاتی خوب برای فراخوانی و همکاری با دیگر برنامه ها. بنابراین در هنگام شروع پیتون یک سوال بالاتر از همه مسائل برای من مطرح بود : این زبان چه چیزی دارد که پرل ندارد ؟
البته ، پرل گوریل 800 کیلویی زبانهای اسکریپت امروزی است. این زبان تا حدی به لطف کتابخانه های یونیکس و اعلانهای جامع خود و تا حدی به دلیل مجموعه بزرگ ماژولهای به وجود آمده توسط جامعه برنامه نویسان پرل ، به نسبت زیادی جایگزین زبانهای اسکریپت انتخابی مدیران وناظران سیستمها شده است. این زبان ، زبان پنهان در پشت حدود 85% محتوای " زنده " بر روی شبکه به شمار میرود. CGI به درستی یکی از مهمترین رهبران جامعه متن باز به ، Larry Wall ، سازنده آن در مکان Richard Stallaman و Linus Torvalds شمار میرود و اغلب پس از سوم جایگاه خدایان قرار میگیرد.
در آن زمان من پرل را برای تعدادی از پروژه های کوچک استفاده کرده بودم. من این زبان را بسیار قدرتمند یافتم گرچه به نظر من میرسید سینتکس و بعضی جنبه های دیگر این زبان میتوانند تا حدی برای شخصی که به آن عادت نداشته باشد خطرناک باشند. به نظر من می رسید که پیتون به عنوان یک زبان اسکریپت باید گام مهم دیگری بردارد ، بنابراین درحال خواندن به دنبال چیزی می گشتم که به نظر برسد آن را از زبان پرل مجزا می کند.
من به سرعت مطالب غیرعادی اصلی ای که ممکن است در پیتون به نظر هر کسی در سینتکس این (indentation) برسد را مرور کردم : این واقعیت که فضای خالی و پرل ندارد، در C زبان بامعنی است ؛ این زبان هیچ سنخیتی با ساختار آکولادی محدودیت گروههای عبارات را از بین (indentation) عوض تغییرات در توگذاری میبرند؛ و مثل اکثر هکرها پس از فهمیدن این مطلب با بی علاقگی عقب کشیدم.

برنامه FORTRAN آنقدر از عمر من گذشته است که برای چند ماهی در دهه 70 با نویسی کرده باشم. اکثر هکرهای این دوره چنین نیستند ولی به نظر می رسد فرهنگ ما به گونه ای ، خاطره ای دقیق وسنتی از اینکه آن زبانهای قدیمی با محدوده ثابت چقدر عذاب آور بودند حفظ کرده باشد. "فرمت آزاد " ، که درآن زمان برای توصیف و پاسکال استفاده می شد C زبانهای token-oriented حالت جدیدتر سینتکس تقریبا فراموش شده است؛ چندین دهه است تمامی زبانها اینگونه طراحی شده اند ، یا تقریبا تمام زبانها. بسیار سخت است کسی را ، پس از دیدن این ویژگی پیتون ، به خاطر عکس العمل اولیه ای مانند اینکه به صورت غیرمنتظره در انبوهی از بی ادبی! .داینا‌سور قدم گذاشته است سرزنش کرد

این احساس من در آن زمان بود. من بدون علاقه چندانی از کنار بقیه توضیحات این زبان گذاشتم. من دلیل چندان دیگری برای توصیه پیتون پیدا نکردم به جز اینکه احتمالا سینتکس آن تا حدی شسته و رفته تر از پرل بود و امکانات موجود برای های ابتدایی مانند دکمه ها ومنوها بسیار خوب به نظر میرسید CGI اجرای .GUI

من کتاب را با این یادآوری ذهنی که باید زمانی یک پروژه کوچک با محوریت انجام دهم ، تا مطمئن شوم که واقعا این زبان را میفهمم ، در قفسه گذاشتم. ولی .باور نداشتم چیزی که دیده ام هرگز بتواند به طور موثر با پرل رقابت کند

اتفاقات زیادی باعث شدند این یادآوری ذهنی ماهها در فهرست اولویتهای من در مکان آخر قرار بگیرد. بقیه سال 1997 برای من پر از اتفاقات بود ، در کنار بقیه اتفاقات ، این سالی بود که من نسخه اصلی "کلیسای اعظم و بازار" را نوشته و منتشر کردم. در عین حال من فرصت یافتم تا چندین برنامه پرل بنویسم ، از جمله برنامه دستیاری ، keeper ، دو برنامه با حجم و پیچیدگی قابل توجه. یکی از آنها است که همچنان برای بایگانی پرونده های ارسالی وارده در شرکت نرم افزاری Metalab به کار میرود. این برنامه تولیدکننده صفحات وبی است که شما در http://metalab.unc.edu/pub/Linux/!INDEX.html ، مشاهده می کنید. برنامه دیگر anthologize ، برای ششمین ویرایش لینوکس از PostScript با هدف تولید خودکار آرشیو " چگونه ها" ی پروژه مستندسازی لینوکس استفاده شد. هر دوی این برنامه ها .موجود هستند Metalab در

نوشتن هر دوی این پروژه ها مرا بیشتر و بیشتر از پرل ناراضی کرد. به نظر میرسید پروژه های با اندازه بزرگتر یک نارضایتی ساده از پرل را تبدیل به مشکلات جدی و مداوم مینمایند. سینتکسی که در صد خط ابتدایی بسیار راضی کننده به نظر میرسید با گذشت زمان تبدیل به توده ای از خارهای غیرقابل نفوذ میشد. " بیش از یک راه برای انجام کار " که حال و هوا و تاثیرگذاری خاصی به آن می بخشید ، نگهداری یک استیل خاص در یک محدوده کد وسیعتر را به مقدار قابل توجهی مشکل می ساخت. وبسیاری قابلیتهایی که بعدا به منظور جوابگویی به نیاز برای کنترل پیچیدگی در و ...) به پرل اضافه شده use strict ، lexical scoping ، برنامه های بزرگتر (اشیا .بودند حالتی شکننده و نه چندان دلچسب داشتند

تمامی این مشکلات دست به دست یکدیگر داده بودند تا خواندن و درک حجم زیادی از کدهای پرل به عنوان یک کلیت را تنها پس از چند روز به صورت غیرمنطقی ای مشکل کنند. علاوه بر این ، من دریافتم که روز به روز زمان بیشتری را به جای ور رفتن با برنامه خود ، صرف ور رفتن با قابلیتهای این زبان میکنم. و بدتر از همه اینکه کد من زشت شده بود ، ... این مساله‌ء واقعی بود. برنامه های زشت مثل پلهای معلق زشت هستند، واژگون شدن آنها بسیار محتمل تر از انواع زیبای آنها است ، زیرا نحوه ادراک انسانها (به خصوص مهندس- انسانها) از زیبایی رابطه بسیار نزدیکی با قابلیت ما برای پردازش و فهم پیچیدگی دارد. زبانی که کار نوشتن کد

.باوقار و شکوه را سخت کند در حقیقت نوشتن کد خوب را مشکل کرده است
با وجود یک دو جین زبان که به آنها تسلط داشتم ، من میتوانستم به تمامی علامات مشهود یک طراحی زبان که به منتهای قابلیت حود رسیده است و چیز بیشتری برای عرضه ندارد پی ببرم. من فکر کردم "باید راه بهتری وجود داشته باشد" و شروع به .گشتن به دنبال یک زبان اسکریپت باشکوه تر کردم
به عنوان یک زبان پیش فرض C موردی که من حتی به آن فکر نکردم بازگشت به بود. دوره ای که انجام مدیریت حافظه در یک برنامه به صورت شخصی معنا داشت ، ، (kernel hacking) به جز در بعضی زمینه ها مانند طراحی هسته سیستم عامل برنامه نویسی علمی و گرافیک 3 بعدی که در آنها شما نیاز دارید بیشترین سرعت و کنترل کامل بر روی استفاده از حافظه به منظور حداکثر استفاده از سخت افزار را .داشته باشید ،مدتهاست که گذشته است
کردن [چندین اصطلاح را در این debug در اکثر موارد دیگر پذیرفتن بار اضافه قسمت حذف کردم م.] و دیگر مشکلات از این دست در ماشینهای امروزی دیوانگی است. بسیار عاقلانه تر است که چندین چرخه و چندین کیلوبایت حافظه برای نیازهای مدیر حافظه یک زبان اسکریپت فدا کنیم و در وقت گرانبهای انسان صرفه جویی کنیم. در حقیقت استفاده از این راهبرد (استراتژی) دقیقا همان چیزی است که .باعث رشد انفجاری پرل از اوایل دهه 90 شده است
ور رفتم تا متوجه شدم که در مقیاسهای بالا حتی از پرل هم ضعیفتر Tcl مدتی با همچنین نگاهی به اشکال ، LISP عمل میکند. به عنوان یک برنامه نویس قدیمی انداختم ولی همانگونه که از نظر Scheme و LISP مختلف رایج (dialects) معمول است ، طراحیهای هوشمندانه متعدد آن به دلیل عدم LISP تاریخی در مورد و یک POSIX/UNIX وجود یا ناقص بودن مستندات ، دسترسی ناقص به امکانات جامعه کوچک و در عین حال بسیار پراکنده کاربران ، بی استفاده شده بودند.
پرطرفدار بودن پرل به هیچ عنوان یک تصادف نیست ؛ اکثر رقبای آن یا برای پروژه های بزرگ بدتر از پرل هستند یا اصلا و ابدا به آن مفیدی که ساختار نظری برتر آنها .انتظارش را ایجاد میکند نیستند
نگاه دومی من به پیتون تقریبا همانقدر تصادفی بود که نگاه اول. در اکتبر 1997 ، یک fetchmail سری سوالات در فهرست نامه های الکترونیک افراد استفاده کننده از این موضوع را کاملا مشخص کرد که افراد برای ایجاد فایلهای تنظیماتِ برنامه دچار مشکلات فزاینده ای میشوند. این فایل دارای ساختاری ساده و fetchmail کلاس به سبک فرمت آزادِ یونیکس است ولی میتواند هنگامی که کاربر دارای چندین باشد به شدت پیچیده شود. به عنوان نمونه ، برای IMAP و POP3 حساب از نوع .دیدن یک مثال ساده شده از برنامه من به لیست 1 نگاه کنید
من تصمیم گرفتم برای حل مشکل از یک ویرایشگر تنظیمات با حالتی دوستانه برای استفاده کنم. هدف از fetchmailconf به نام (End-User-Friendly) کاربر نهایی مشخص بود ؛ پنهان کردن کامل سینتکس فایل کنترل در پشت fetchmail طراحی با کلیدهای (GUI) ظاهر شیک و صحیح از نظر ارگونومی یک رابط گرافیکی کاربر .انتخاب ، منوهای کشویی و فرمهای مختلف
را در GUI فکر پیاده کردن این برنامه با پرل چندان مرا ذوق زده نکرد. من کدهای که حتی از کد پرل خود برنامه Tcl پرل دیده بودم که ترکیبی ناهمگون بود از پرل و هم زشت تر بود. در این لحظه بود که قولی را که 6 ماه قبل داده بودم یادم آمد. این .میتوانست فرصتی برای کسب یک تجربه عملی با پیتون باشد
البته این موضوع مجددا من را با بامعنی بودن فضای خالی در پیتون روبرو کرد. البته نوشتم. جداً GUI این بار تخته گاز پیش رفتم و مقداری کد برای انبوهی از عناصر

حیرت آور بود که تنها پس از 20 دقیق ، استفاده پیتون از فضای سفید دیگر امر عجیبی به نظر نمی آمد. من همانقدر از فرورفتگی متن در پیتون استفاده میکردم که از آن استفاده کنم و این روش جواب هم میداد C امکان داشت در.
این اولین حیرت من بود. حیرت دومی ، حدود 2 ساعت بعد (با احتساب زمانهای صرف Learning **Python** توقفی که برای نگاه کردن به قابلیتهای پیتون در کتاب شد) ، وقتی پیش آمد که متوجه شدم با همان سرعتی مشغول نوشتن برنامه های به دردبخور هستم که میتوانم تایپ کنم. وقتی متوجه این موضوع شدم تقریبا مبهوت شدم. یکی از معیارهای مهم عملکرد در برنامه نویسی تعداد دفعاتی است که شما چیزی می نویسید که واقعا با درک شما از مساله همخوانی ندارد و ناچارید پس از فهمیدن این موضوع که چیزی که تایپ کرده اید به زبان آنچه را شما فکر می کنید نمی گوید به عقب برگردید. یکی از معیارهای مهم طراحی مناسب زبان اینست که درصد چنین اشتباهاتی با تجربه پیدا کردن شخص در آن زبان با چه سرعتی کاهش می یابند.
هنگامی که شما با بیشترین سرعتی که قادرید تایپ کنید برنامه به دردبخور می نویسید و اشتباهات شما در انتخاب مسیر تقریبا صفر است ، این به آن معناست که شما در آن زبان به استادی رسیده اید. ولی این بی معنی بود ، چون به زحمت یک روز از این موضوع گذشته بود و من هنوز مجبور بودم امکانات مربوط به زبان و کتابخانه ها را از روی کتاب نگاه کنم.
این اولین چیزی بود که به من فهماند من در پیتون با یک طراحی عالی سروکار دارم. اکثر زبانها آنقدر دارای تداخل و ضعف در طراحی خود هستند که شما قبل از اینکه ضریب اشتباهاتتان در آنها حتی نزدیک به صفر شود بسیاری از امکانات آنها را فرا گرفته اید. پیتون تنها زبان همه منظوره ای بود که من تا آن روز استفاده کرده بودم و این فرآیند را معکوس می کرد.
البته یادگیری مجموعه امکانات آن نیز چندان وقت گیر نبود. من یک برنامه را در 6 روز کاری نوشتم که ، GUI کاراً و قابل استفاده ، به همراه fetchmailconf از این 6 روز احتمالا 2 روز آن تماما صرف یادگیری خود پیتون شد. این موضوع نشان دهنده یک خاصیت مفید دیگر این زبان است : این زبان حالت اختصاری دارد ، شما میتوانید تمامی مجموعه امکانات آن (حداقل یک ایده فهرست وار از کتابخانه های آن زبانی است که به اختصار معروف است. پرل در این زمینه C .) را در ذهن نگه دارید شهرت خوبی ندارد. یکی از چیزهایی که عبارت معروف " بیش از یک راه برای انجام آن وجود دارد! " قربانی می کند توانایی مختصر بودن پرل است.
ولی هنوز رویایی ترین کشف من باقی مانده بود. طراحی من یک مشکل داشت : من کاربر تولید کنم ِ GUI قادر بودم به راحتی فایلهای تنظیم را با استفاده از عملهای ولی ویرایش آنها مشکل بزرگتری بود. یا بهتر است بگویم خواندن آنها به یک شکل قابل ویرایش کار مشکلی بود.
بسیار کاربر بود. این fetchmailبه کار رفته برای سینتکس فایل تنظیمات Parser دو ابزار کلاسیک یونیکس برای تولید ، Lex و YACC برنامه در حقیقت با استفاده از نوشته شده بود. تصور من این بود که برای اینکه قادر به ، C زبان در parsing کد وقت ِ parser باشم ناچار به تکرار همان fetchmail ویرایش فایلهای تنظیمات در گیر در پیتون خواهم بود. من برای انجام این کار بسیار دودل بودم ، تا حدودی به دلیل حجم کار و تا حدودی هم به دلیل اینکه نمیدانستم از کجا باید مطمئن باشم که دو در دو زبان مختلف مثل یکدیگر عمل خواهند کرد. کار فوق العاده زیاد parser آخرین چیزی بود که ممکن بود من با تحول زبانها به parse همخوان نگاه داشتن دو آن نیاز داشته باشم.

این مشکل برای مدتی مرا متوقف کرد. بعد از این فکری به ذهنم رسید : من به fetchmailconf اجازه می دادم از parser خود استفاده کند. من یک گزینه – اضافه کردم و نتیجه را در یک خروجی استاندارد با fetchmail به configdump پیتون تخلیه کردم. برای فایل بالایی ، نتیجه تقریبا شبیه به initializer فرمت یک لیست 2 بود (برای صرفه جویی در فضا بعضی داده های غیرمرتبط با مثال حذف شده اند).
را ارزیابی کند و سپس fetchmail--configdump پیتون سپس قادربود خروجی تنظیمات را به صورت مقدار متغیر " fetchmail " ارائه کند.
حتی این هم آخرین حرکت رقص من نبود. چیزی که من میخواستم صرفا این نبود که fetchmail تنظیمات فعلی خود را داشته باشد بلکه میخواستم آن را تبدیل به یک کنم. احتمالا در این درخت سه نوع شی (live objects) ساختار درختی از اشیا زنده وجود داشت : تنظیمات (که شیء واقع در بالاترین مرتبه و نشان دهنده کل تنظیمات بود) ؛ سایت (نشان دهنده یکی از سایتهایی که باید به رای گذاشته می‌شد.) و کاربر (نشان دهنده داده‌های کاربران که به سایت متصل می‌شود). فایلهای مثال پنج شیء از نوع سایت که به هر کدام یک شیء از نوع کاربر متصل شده است را نشان می‌دهد.
تا اینجا من سه شیء کلاس را طراحی کرده و نوشته بودم (این همان چیزی بود که چهار روز از وقت من را گرفت که اکثر آن هم صرف چیدن درست و مناسب کنترلهای صفحه شد). هر یک از این کلاسها یک متد داشت که باعث می شد برای باز شود. تنها مشکل باقیمانده GUI خاص یک صفحه instance ویرایش داده های آن پیتون به اشیاء زنده initializer من یافتن روشی برای تبدیل اشیاء مرده این برنامه بود.
چیزی که به نظر من رسید نوشتن کدی بود که به وضوح راجع به ساختار هر سه و ایجاد اشیایی که پاسخگوی initializer کلاس بداند و از این دانش برای پلکیدن در نیازها باشند استفاده کند ولی این ایده را رد کردم زیرا اعضای این کلاس جدید احتمالا با اضافه شدن امکانات جدید به تنظیمات زیاد می شدند. اگر من کد ایجاد اشیا را با این روش مشهود می نوشتم ، احتمال زیادی وجود داشت که کد حالت از حالت به روز initializer شکننده پیدا کند و با تغییر تعریفهای کلاس یا ساختار خارج شود.
را تجزیه initializer چیزی که من واقعا به دنبالش بودم کدی بود که شکل و اعضای و تحلیل کند ، از تعریفهای کلاس برای پرس و جو درباره اعضای آن کلاس استفاده کند و سپس خودش را با این دو مجموعه هماهنگ کند.
گفته می شود و در حالت metaclass به این گونه چیزها در اصطلاح هک کردن عمومی چیزی به ترسناکی جادوی سیاه به شمار میروند. اکثر زبانهای برنامه نویسی شیءگرا مطلقا از این قابلیت پشتیبانی نمی کنند و در آنهایی که این کار را می کنند (که پرل هم یکی از آنهاست) ، این موضوع امری پیچیده و ظریف و شکننده است. تا همان زمان من از ضریب بسیار پایین اشکال سازی پیتون متعجب شده بودم ولی این یک آزمایش واقعی بود. برای اینکه این زبان را مجبور به انجام چنین کاری کنم چقدر باید با آن کشتی می گرفتم ؟ با استفاده از تجربیات قبلی می دانستم که ، حتی با این فرض که من ببرم ، این می تواند تجربه دردناکی باشد ولی روی کتاب شیرجه پیتون کردم. تابع به دست آمده در metaclass رفتم و شروع به مطالعه امکانات فهرست 3 نشان داده شده است و کدی که فراخوانی می کند در فهرست 4 آمده است.
برای جادوی سیاه چندان بد به نظر نمی رسد ، اینطور نیست ؟ سی و دو خط با احتساب توضیحات. فقط با توجه به چیزهایی که من درباره ساختار کلاسها گفتم ،

حتی کد فراخوان نیز قابل خواندن است. ولی اندازه این کد موضوع شوکه کننده نیست. خودتان را کنترل کنید : نوشتن این کد تنها 90 دقیقه وقت گرفت و در اولین باری که من آن را اجرا کردم به خوبی کار کرد.

تنها گفتن این که من حیرت زده شده بودم می تواند کم لطفی باشد. درست و به خوبی کار کردن پیاده سازی حتی ساده ترین تکنیکها در اولین بار همانگونه که انتظار من تنها metaclass می رفته میتواند امری قابل توجه باشد ؛ ولی اولین هک کردن پس از 6 روز از یک شروع بی مقدمه ؟ حتی اگر به اغراق بگوییم که من یک هکر با استعداد هستم این یک موفقیت بزرگ برای پیتون در زمینه شفافیت و شکوه و عظمت طراحی است.

تقریبا هیچ راهی وجود نداشت که من بتوانم با وجود تجربه بسیار زیادترم در پرل عمل مشابهی را با استفاده از آن انجام دهم. در این زمان بود که احساس کردم که احتمالا پرل را ترک خواهم کرد.

این رویایی ترین لحظه من در پیتون بود. ولی حالا که همه اینها تمام شده باید بگویم تمام اینها یک هک هوشمندانه بود. مفید بودن یک زبان برنامه نویسی در بلند مدت بستگی به پشتیبانی آنها از هک های هوشمندانه ندارد بلکه به چند و چون پشتیبانی آن از امور روزمره برنامه نویسی مرتبط است. کار روزمره برنامه نویسی اکثرا به جای نوشتن برنامه های جدید ، از خواندن و تغییر دادن برنامه های موجود تشکیل می شود.

من ، fetchmailconf مساله تکان دهنده اینجاست : هفته ها و ماهها پس از نوشتن هنوز قادر بودم کد آن را بخوانم و بدون هیچ تلاش فکری جدی ای متوجه بشوم که چکار میکند. و دلیل واقعی این موضوع که من دیگر برای هیچ چیزی جز پروژه های کوچک از پرل استفاده نمی کنم این است که این موضوع هرگز در مورد پرل صادق نبود. من حتی از تصور این موضوع که زمانی ناچار شوم anthologize یا keepr را تغییر دهم به وحشت می افتم در حالیکه برای انجام چنین کاری در مورد fetchmailconf ککم هم نمیگزد.

پرل همچنان کاربردهای خود را دارد. برای پروژه های کوچک (100 خط یا کمتر) که شامل مقدار زیادی جور کردن الگوی متن است ، من احتمالا بیشتر از یک راه حل با استفاده از پرل استقبال میکنم تا پیتون. برای دیدن چندین مثال خوب ، برنامه های مشاهده کنید. در حقیقت fetchmail را در فهرست توزیع growplot و timeseries ، اینها تا حد زیادی شبیه کاری هستند که پرل در نقش اصلی خود ، قبل از اینکه سیستم عامل باشد ، به عنوان نوعی ِAPI دارای توابع و دسترسی مستقیم به انجام می داد. برای هر چیز بزرگتر یا پیچیده تر ، من awk/sed/grep/sh ترکیب عادت کرده ام که ارزشهای لطیف پیتون را ترجیح بدهم و فکر کنم شما نیز همین کار را انجام بدهید.