

The PCLinuxOS magazine

Volume 149

June, 2019



De-Googling Yourself – Part 2

***Forgotten Wifi Passwords? Here's
How To Find Them***

Gmail Eerily Tracks Your Purchases

***Short Topix: Google, Other Tech
Giants Buying Up Internet Undersea
Cables***

***PCLinuxOS Friends & Family
Spotlight: BillDjr***

***PCLinuxOS Recipe Corner: Crispy
Whole Chicken & Vegetables***

Inkscape Tutorial: Creating An Avatar

***Ruby Programming Language: Data
Handling with Variables, Classes,
Arrays and Strings***

***ms_meme's Nook:
Black Painted Desktop***

Casual Python – Part 5

And more inside ...

In This Issue ...

- 3 From The Chief Editor's Desk ...**
- 4 Casual Python, Part 5**
- 12 Screenshot Showcase**
- 13 ms_meme's Nook: Black Painted Desktop**
- 14 PCLinuxOS Recipe Corner:
Crispy Whole Chicken & Vegetables**
- 15 Inkscape Tutorial: Create An Avatar**
- 17 Screenshot Showcase**
- 18 De-Googling Yourself, Part 2**
- 23 Gmail Eerily Tracks Your Purchases**
- 26 Screenshot Showcase**
- 27 The Ruby Programming Language:
Data Handling with Variables, Classes, Arrays and Strings**
- 37 Forgotten Wifi Passwords? Here's How To Find Them**
- 39 Screenshot Showcase**
- 40 Short Topix: Google, Other Tech Giants
Buying Up Internet Undersea Cables**
- 43 Screenshot Showcase**
- 44 PCLinuxOS Family Member Spotlight: BillDjr**
- 46 PCLinuxOS Bonus Recipe Corner:
5-Ingredient InstaPot™ Barbecue Pork Ribs**
- 47 ms_meme's Nook: PCLOS Flyer**
- 48 Screenshot Showcase**
- 49 PCLinuxOS Puzzled Partitions**
- 53 More Screenshot Showcase**

The **PCLinuxOS** magazine

The PCLinuxOS name, logo and colors are the trademark of Texstar.

The PCLinuxOS Magazine is a monthly online publication containing PCLinuxOS-related materials. It is published primarily for members of the PCLinuxOS community. The magazine staff is comprised of volunteers from the PCLinuxOS community.

Visit us online at <http://www.pclosmag.com>

This release was made possible by the following volunteers:

Chief Editor: Paul Arnote (parnote)

Assistant Editor: Meemaw

Artwork: ms_meme, Meemaw

Magazine Layout: Paul Arnote, Meemaw, ms_meme

HTML Layout: YouCanToo

Staff:

ms_meme

Meemaw

Gary L. Ratliff, Sr.

Daniel Meiß-Wilhelm

daiashi

Cg_Boy

YouCanToo

Pete Kelly

Smileeb

Alessandro Ebersol

Contributors:

BillDjr

The PCLinuxOS Magazine is released under the Creative Commons Attribution-NonCommercial-Share-Alike 3.0 Unported license. Some rights are reserved.
Copyright © 2019.



From The Chief Editor's Desk ...

Phew! Has it ever been a busy past couple of months. I **finally** got my failing home central air conditioner replaced. I went ahead and replaced my home's furnace as well, at the same time. The AC unit was 21 years old (and failing miserably). The new furnace replaced my 18 year old furnace. The two together should save me considerable money in utility bills. Plus, they are a matched pair. Where I

stay home and help take care of the kids and helping tend to my wife's needs as she recovers. She's off work for six weeks total, and is already seeing some positive benefits of having the surgery.

I've been using some of the extra time off from work to also get some much needed work done around the house. I've performed some minor plumbing

repairs, put new rain gutter guards on the rain gutters (and cleaned out the gutters, while I was at it), mowed the grass a couple of times, as well as cooking up some meals for me and the kids (since my wife can't start to eat anything solid for four weeks after the surgery). Of course, thrown into the mix is getting some work done for the magazine.

Also, around my area of the U.S., we've been dealing with some serious flooding. Many of the reservoirs are full, and the U.S. Army Corps of Engineers is looking to have to release some water downstream to preserve the integrity of the reservoirs. Of grave concern is the additional flooding downstream that it might cause. We have endured one of the wettest months of May on record. The streams and rivers are already running at flood stage levels, if not already over flood stage. In southeast Kansas, where Meemaw lives, the

flooding is especially bad. The walking trail she typically uses for her evening walks is currently under six feet of flood water. At one point, she couldn't even get into the nearest town because the highway was under flood waters. Those waters have since receded, but it could happen again if we continue to get additional rainfall, or if the U.S. Army of Corps Engineers is forced to release more water from the reservoirs.

This Month's Cover

June 14 in the U.S. is celebrated as [Flag Day](#). It's a day when Americans celebrate the adoption of the U.S. flag by the 1777 Continental Congress, the flag's history, and to show appreciation and pride for the flag. I thought why not expand it so that everyone, everywhere could celebrate their appreciation and pride in their flag. So, this month's cover includes the flags of 207 countries. According to [WorldAtlas.com](#), there are 197 countries in the world (the U.N. only recognizes 195, refusing to recognize Taiwan and Kosovo), so I'm not exactly sure where the extra 10 countries come from. At first, I thought that they might have included the flags of countries that no longer exist, like the USSR and East Germany. But upon closer inspection, I discovered that those flags are not included. The image shows the flags of Palestine and the EU (which is a collection of sovereign countries), as well as other countries not formally recognized by a variety of agencies or governments. Of course, that left two blank spots in the image. I couldn't have that, so I added in one for PCLinuxOS, and another for Linux (consisting of Tux).



live, near Kansas City, it gets hot and humid in the summer (usually 80% relative humidity and higher, with temperatures 80-100° F), and it gets quite cold in the winter (typically with temperatures 20-40° F, but it's not uncommon for the temperatures to drop to as low as sub-zero° F when an arctic cold front moves through).

Meanwhile, my wife has been recovering at home after having surgery to have her gastric lap band removed, and having the gastric sleeve performed. I've taken about a week and a half off from work to



Casual Python, Part 5

by critter

Docz revisited again

Unfortunately, this is a regular pattern in application programming. If you built and tried out the docz application, you may have noticed a couple of bugs in there. It seemed to work fine, but then one day you happen to click on the first item and it tells you that it doesn't exist. How can this be? The application found the name, but you didn't type it in. Then you find another problem. If a file name has spaces in it, the application tries (and fails) to open multiple documents.

The first problem is because I made an assumption – that the first character always had to be removed. It appeared to work because I never needed to select the first item which, now it seems, doesn't have a 'bad' character at the beginning. What is returned in resultslist is a list of strings separated by a strange character (actually a multi-byte Unicode character known as a paragraph separator – ord(8233)). As this is a separator, it is only present at the beginning of the second and subsequent strings in the list, so we can test for its presence and if found, then, and only then, remove it.

The second one is easy to fix. If the filename has spaces, then what is passed to the subprocess command is a list of arguments, each one it believes to be a document to open. The solution is to wrap the filename in quote marks. As single quotes are used in the subprocess command the new quotes must be double quotes.

Here is the revised method code:

```
def mousePressed(self, Event):
    textCursor = self.resultslist.cursorForPosition(Event.pos())
    textCursor.select(QTextCursor.BlockUnderCursor)
    self.resultslist.setTextCursor(textCursor)
    f_name = textCursor.selectedText()
    if f_name == '':
        pass
    else:
        if ord(f_name[0]) == 8233:
            f_name = f_name[1:]
        if ' ' in f_name:
            f_name = '"' + f_name + '"'
        command = ('libreoffice6.0 --writer ' + f_name)
```

```
subprocess.Popen(command, shell=True)
self.exitApplication()
```

More about strings

The strings that I have covered so far have all been common everyday text strings. The sort you type into e-mail messages, or that this document comprises. In these strings, whenever a '\n' character is encountered, the system interprets this as a newline character and accordingly starts a new line of text. The \n character is one of a group of special characters known as control codes, because they control how the system behaves. But what if you actually wanted to output a slash followed immediately by a letter n. How would you do that?

Well one way is to 'escape' the code by preceding it with another slash.

```
s = 'Hello \\nworld!'
print(s) ==> Hello \nworld
```

This can quickly become unwieldy, so python has a way of doing that by prefixing the string with a letter 'r' to denote a 'raw' string. This is of particular use when using regular expressions that expect to see the slash character.

```
s = 'Hello \nworld!'
print(s) ==> Hello
                world!
```

```
s = r'Hello \nworld!'
print(s) ==> Hello \nworld
```

Regular expressions are almost a language of their own, and to use them in python, you must import the re module. One unusual aspect of the re module is that it contains a method that allows you to compile the regular expression (also commonly referred to as a regex). This makes its use so much easier. I shall demonstrate this in a simple little utility called solver.

A string can also be preceded by a 'b' which denotes a bytes string. A bytes string is not usually human readable (and it can be risky to try). It is a sequence of machine readable bytes, and to be of use, the reader has to understand the format. I'll say more about byte strings when we need to use them, but for now I'll stick to text mode.

Solver

My wife enjoys trying to solve crossword puzzles. You know, those blocks of intersecting words often found in newspapers or at the end of the PCLinuxOS magazine. Occasionally, she gets stuck and can go no further without a little help. I wrote this for those situations.

To use it you type in the letters that you know, replacing the unknown letters with a space or a period '.'. The application then tries to find all possible matches from a list of standard words, and then displays them as possible solutions.

Open the synaptic package manager and install 'words'. This will put a list of almost half a million words, each on a separate line, into /usr/share/dict named linux.words.

`wc -l /usr/share/dict/linux.words` currently returns 483523. This will be the reference list.

This application is almost exactly the same as the thesaurus application so we can save ourselves some work by making a copy of that.

Make a new directory and copy over thesaurus.py, renaming it to solver.py, and thesaurus.ui, renaming that to solver.ui. Copy over update_res.sh, and change the contents to:

```
#!/usr/bin/env bash
```

```
pyuic5 solver.ui > solver_ui.py
```

Add a suitable icon to the directory/

Open solver.ui in Designer and make the following changes:

Form

Window title ==> Solver

Window icon ==> whatever icon you chose

clearButton

objectName ==> resetButton

Label

Change the icon to use your icon

Label_2

text == > Enter the word. Replace unknown letters with a '.' or space

X ==> 5

Width ==> 410

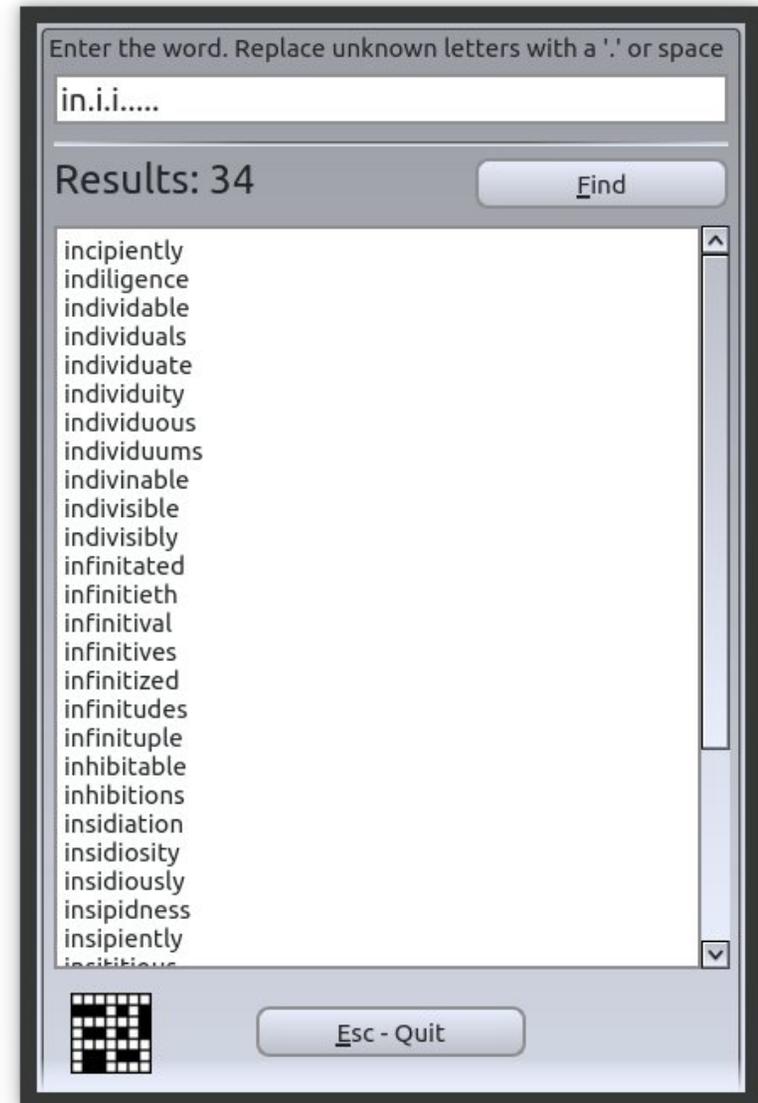
Label_3

objectName ==> resultsLabel

Width ==> 200

Again these are only my suggestions but the objectNames must match what is in the code.

This should now look like the image below.



Execute update_res.sh to generate the new solver_ui.py file.

The code is very similar to thesaurus, with a new `correct_input` method added. Here it is in geany, with the methods closed up for brevity.

```
solver.py
1  #!/usr/bin/env python3
2
3  from PyQt5.QtCore import *
4  from PyQt5.QtWidgets import *
5  import sys
6  import re
7  import solver_ui
8
9
10 class Solver(QWidget, solver_ui.Ui_Form):
11     def __init__(self):
12
13
14     def keyPressEvent(self, e):
15
16
17     def find_it(self):
18
19
20     def correct_input(self):
21
22
23     def reset(self):
24
25
26     def exitApplication(self):
27
28
29 if __name__ == '__main__':
```

We no longer need to import the subprocess module, so that line has been deleted. The `re` import is new. We now import `solver_ui` instead of `thesaurus_ui`, and the `correct_input` method has been added. The class definition is now named `Solver`, and inherits from `solver_ui`.

```
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
import sys
import re
import solver_ui
```

```
class Solver(QWidget, solver_ui.Ui_Form):
```

This is the `__init__` method.

```
def __init__(self):
    super(Solver, self).__init__()
    self.setupUi(self)
```

```
self.textinput.setFocus()
self.textinput.setToolTip(
    'Press the return key to display the result')
self.resetButton.clicked.connect(self.reset)
self.quitButton.clicked.connect(self.exitApplication)
self.textinput.textChanged.connect(self.correct_input)
self.textinput.returnPressed.connect(self.find_it)
```

Only the highlighted line has been added, which connects to a new method if the text in the textinput box has changed. The `clearButton` is now named `resetButton`.

The `correct_input` method looks like this:

```
def correct_input(self):
    line_len = len(self.textinput.text())
    if line_len:
        if self.textinput.text()[line_len - 1] == ' ':
            newtext = self.textinput.text()[:line_len - 1] + '.'
            self.textinput.setText(newtext)
```

This new method sets the variable `linelen` to the length of the text currently in the textinput box. The first `if` statement could be read as 'if there is any text in the input box then do the following'. The second `if` statement checks if the last character is a space, and if so, changes it to a period. This enables the user to use either a space or a period for unknown characters. This method was not really necessary, but adds a little friendliness to the application usage.

The `keyPressEvent`, `reset` and `exitApplication` methods are all unchanged.

The main changes are in the `find_it` method.

```
def find_it(self):
    self.resultslist.clear()
    self.target = self.textinput.text()
    results = ''
    count = 0
    pattern = '^' + self.target + '$'
    regex = re.compile(pattern)
    for word in open('/usr/share/dict/linux.words'):
        bareword = word.rstrip('\n')
        result = regex.search(bareword)
        if result:
            count += 1
            results = results + word
    self.resultslist.setText(results)
    self.resultslabel.setText("Results: " + str(count))
```

The highlighted lines are new, and three lines have been deleted.

The variables 'results' and count are initialised to the empty string "" and 0 respectively. This is necessary, as a variable may not be used until it has been defined. As a variable is really just a name for a reference to an object, the object must exist before the variable can be used.

What we are going to search for is going to be a string referenced by pattern. In a regular expression, the ^ character means 'the beginning' and the \$ character means 'the end', so we have beginningtextinput.textend. In other words, the whole of the contents of the textinput box.

This line:

```
regex = re.compile(pattern)
```

calls the compile method of the re module we imported, and puts the result in to the variable regex. What this variable holds is not a string, but an object of the class

```
re.RE_Pattern.
```

The re module is complex but powerful. The objects it creates are used to get results. They do not hold the results. If you need to do a lot of text processing, then you will almost certainly want this module, but be prepared for a bit of a learning curve.

In the for loop, we open the list of words we installed and strip off the end of line (\n) character of each word. We then ask our compiled regular expression object to use its search method to look for a match. If found, then result is passed as a match object. If not, then it is passed **None**, and the result becomes the empty string "". **None** is another standard python object. It is not zero or the empty string – it simply denotes a lack of value (technically this is known as a singleton).

The if statement checks this, and for every match, increases the count by one, adds the word to the text in the results variable, and the resets the the text in resultslist to this modified text. The value in count is converted to a string and added to the text in resultslabel.

Finally, form is now an instance of class Searcher.

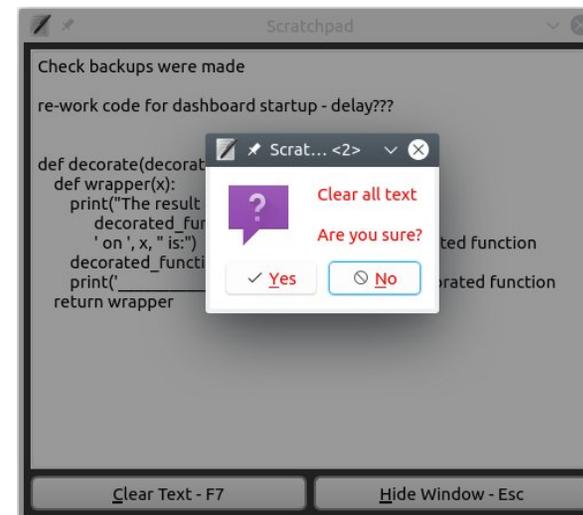
```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = Solver()
    form.show()
    app.exec_()
```

Around 20 lines added or changed for a completely new application.

Scratchpad

This is a little application that sits in the system tray and replaces all of those little sticky notes or the applets that are meant to be a replacement, but for me, were just not what I wanted. This then was an opportunity to create my own version that works as I would like it to.

Scratchpad sits in the system tray, and pops up a window with a left mouse click. Whatever you enter is automatically saved, and pressing the escape key or clicking the hide button tucks it away again. While the window is open, you have the option to clear all the text by a button click or the F7 key, and a confirmation box pops up, are you really sure? Once it has gone, it's gone. Copy and paste is supported (text only), and a right mouse click gives the option to exit or to see an about box – Duh! That's it, but I have found this to be extremely useful.



The main window and confirmation dialog. Note that the dialog is modal. That is, it blocks the parent window until it is closed.



The tray icon and the about box.



Creating the application

Copy `template.ui` to the new directory and rename it to `scratchpad.ui`, then copy `qt5_template.py` over, renaming that to `qt5_scratchpad.py`. Add a suitable icon to the new directory – this is what you will see in the taskbar. Edit `update_res.sh`, changing the word `template` to `scratchpad` in two places.

In designer, change the form to look like the screenshot. I have given the form a width/height of 500,400, changed the size policy to fixed, and given min/max sizes of 500 and 400 the same as the form width/height. I did this because this is a popup application, and I expect to always have the same size window. I have no need to resize it. Fix the form icon, form title and button labels. Remove the central label. Set the textEdit stylesheet to:

```
background-color: white; color:black
```

And the form stylesheet to:

```
background-color: #353836
```

This just gives a bit of contrast. The buttons I resized to 240 x 30.

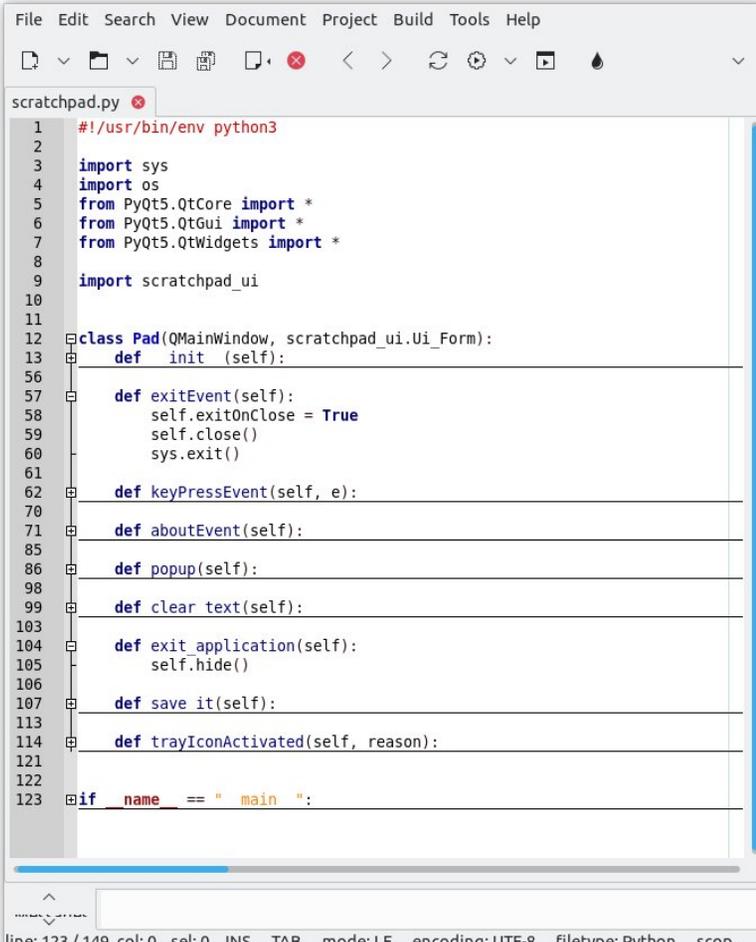
Check it out with `control - r`, save it and run `update_res.sh`. Check that you now have a file named `template_ui.py`, and that is not size zero bytes – mine is 3682 bytes, yours may vary. The user interface is now done.

The code is given below in its collapsed form in geany. I have left two methods open, as these are unusual.

As this is a tray resident application, we want to hide the form, but leave the application running with its icon in the tray. To do this, we connect the `quitButton` press event to the `exit_application` method, which calls `self.hide()`. The `key_press` event also connects the escape key to `self.hide()`. In this state, the application is still running, waiting for events to happen, but our form, which is an instance of this `Pad` class, is hidden. Fully closing down the application is done through a right click menu, which calls the `exitEvent` method. This method does the shutdown procedure. Going through a right-click menu avoids accidental shutdown of the application (right).

I have added a couple of imports at the top of the file:

import os. This gives some operating system specific methods. In this application, it is used to manipulate the file system in the initialization phase.



```

1  #!/usr/bin/env python3
2
3  import sys
4  import os
5  from PyQt5.QtCore import *
6  from PyQt5.QtGui import *
7  from PyQt5.QtWidgets import *
8
9  import scratchpad_ui
10
11
12 class Pad(QMainWindow, scratchpad_ui.Ui_Form):
13     def __init__(self):
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

From `Pyqt5.QtGui import *` This is needed for some Qt graphical elements we have not used before, specifically **Qicon** and **Qpixmap**.

Our user interface import is now

```
import scratchpad_ui
```

The class definition is now

```
class Pad(QWidget, scratchpad_ui.Ui_Form):
```

Most of the work is in the `__init__` method that looks like this:

```
def __init__(self):
    super(self.__class__, self).__init__()
    self.setWindowFlags(Qt.Tool)
    self.setupUi(self)
    self.textEdit.setFocus()

    self.homedir = os.environ['HOME']
    # create the scratchfile if it does not exist
    self.scratchfile = self.homedir + '/.scratch'
    if not os.path.exists(self.scratchfile):
        os.mknod(self.scratchfile)
    self.in_file = open(self.scratchfile, 'r')
    self.textEdit.setText(self.in_file.read())

    # override the close event
    self.exitOnClose = False

    # implement the right-click menu
    my_exit = QAction("Exit", self)
    my_exit.triggered.connect(self.exitEvent)
    my_about = QAction("About", self)
    my_about.triggered.connect(self.aboutEvent)
    menu = QMenu(self)
    menu.addAction(my_exit)
    menu.addAction(my_about)

    # set the application and tray icons
    icon = QIcon()
    icon.addPixmap(QPixmap("scratchpad.png"),
                  QIcon.Normal, QIcon.Off)
    self.tray_icon = QSystemTrayIcon()
    self.tray_icon.setIcon(QIcon(icon))
    self.setWindowIcon(QIcon(icon))
    self.tray_icon.setContextMenu(menu)
    self.tray_icon.setToolTip('Enter text as quick notes or
reminders')

    # Left click event
    self.tray_icon.activated.connect(self.trayIconActivated)

    self.tray_icon.show()
    self.clearButton.clicked.connect(self.clear_text)
    self.quitButton.clicked.connect(self.exit_application)
    self.textEdit.textChanged.connect(self.save_it)
```

In the first block of code, we set the window flags attribute to `Qt.Tool`. This is a minimal type of window that has no maximize button. After `setupUI` has initialized,

the form the `textEdit` window is given the focus.

The next block uses the `os` modules `environ` method to find the users home directory. It then adds `./.scratch` to this to provide a fully qualified filename for a hidden file in the users home directory. This ensures that each user has their own `.scratch` data file. If this file does not exist, then it is created. The file is opened and any text it contains is put into the `textedit` window.

We then set the `exitOnClose` attribute to `False`. An attribute is a methods variable value, and `False` is a python Boolean value. What this means is that if we click on the close button of the window, it does not close. This is known as overriding the default method and means that we control when the application will be closed. We take control.

Next, we implement the right click menu. We have two actions named `my_exit` and `my_about`, and these are given the text that will appear in the menu that pops up when we right click on the icon in the tray. These in turn are connected to methods that will be executed when the item is clicked. We then give the name menu to a `Qmenu` object that will be created for us, and the actions we created are added to this menu. This means that we create items that will appear in the menu, and direct any click events to the appropriate action.

We then create a `QIcon` object named `icon`, and tell it to use our icon image.

A `QSystemTrayIcon` object is given the name `tray_icon`, the icon it will use is set to our icon. The window icon is set to the same icon, and our context menu that we previously created is added to the tray icon. Then we add a tooltip that will be shown when the mouse hovers over the tray icon.

The next thing to do in the initialization phase is to set up the left click event – what to do when the tray icon receives a left mouse click. We connect the event to the `trayIconEvent` method, which we will define shortly.

The `exitEvent` is almost the same code we usually use to exit an application, with the exception that we first have to set the `exitOnClose` attribute to `True` (remember that we initially set this to `False`).

```
def exitEvent(self):
    self.exitOnClose = True
    self.close()
    sys.exit()
```

The `keyPressEvent` method intercepts the escape key even and hides the application form. The `F7` keypress calls the `clear_text` method which I shall explain in a moment.

```
def keyPressEvent(self, e):
    if e.key() == Qt.Key_Escape:
        self.hide()
    if e.key() == Qt.Key_F7:
        self.clear_text()
```

Next we define two popup dialogs. There are two ways to create a popup dialog in Qt5. The method shown below is the long way, but gives you more control over the contents. The other way is quicker, and usually sufficient. I will use the second method in the next example, an alarm/timer utility.

The aboutEvent method sets up a QMessageBox we assign to the variable mbox. The popup method defines one that we give to the variable popup. These are very similar, and the code is almost self explanatory. We fill in some detail such as:

what text to display

which icon to display – you can choose from:

	Question	For asking a question during normal operations.
	Information	For reporting information about normal operations.
	Warning	For reporting non-critical errors.
	Critical	For reporting critical errors.

The actual icon displayed depends on the system icon theme.

What buttons to display and which will be the default. The choices are:

OK, Open, Save, Cancel, Close, Yes, No, Abort, Retry and Ignore.

Not all are always available or make sense.

The QMessageBox returns a value from its exec_ method (note the trailing underscore, this keeps it separate from python's own exec builtin). It is this value that we used in the keyPressEvent method for the answer.

```
def aboutEvent(self):
    mbox = QMessageBox()
    mbox.setIcon(QMessageBox.Information)
    mbox.setWindowTitle("About")
```

```
mbox.setStyleSheet("background:white; color:black")
mbox.setText("Qt5 Scratchpad")
msgText = (" A simple note taking \n" +
           "tray resident application\n" +
           " Programmed in Python3\n" +
           " and PyQt5\n")
mbox.setInformativeText(msgText)
mbox.setStandardButtons(QMessageBox.Ok)
mbox.setDefaultButton(QMessageBox.Ok)
mbox.exec_()
```

```
def popup(self):
    popup = QMessageBox()
    popup.setIcon(QMessageBox.Question)
    popup.setText("Clear all text")
    popup.setStyleSheet("background:white; color:red")
    popup.setWindowTitle("Scratchpad")
    popup.setInformativeText("Are you sure?")
    popup.setStandardButtons(QMessageBox.Yes |
```

```
QMessageBox.No)
    popup.setDefaultButton(QMessageBox.No)
    answer = popup.exec_()
    return answer
```

The clear_text method is fairly straightforward.

```
def clear_text(self):
    answer = self.popup()
    if answer == QMessageBox.Yes:
        self.textEdit.setText('')
```

Our popup messagebox is displayed, which returns the value of the button pressed. If the yes button was pressed, then the text is cleared. Otherwise, no action is taken.

The save_it method does what you would expect.

```
def save_it(self):
    file_out = self.scratchfile
    out_file = open(file_out, 'w')
    text = self.textEdit.toPlainText()
    out_file.write(text)
    out_file.close()
```

The hidden .scratch file is opened in write mode, the text is converted to plain text (drag'n drop can sometimes leave something weird in there). This is then written to the file and the file closed.

The `trayIconActivated` method monitors the icon in the tray for mouse clicks.

```
def trayIconActivated(self, reason):
    if reason == QSystemTrayIcon.Context:
        self.tray_icon.contextMenu().show()
    elif reason == QSystemTrayIcon.Trigger:
        self.show()
        self.raise_()
        self.activateWindow()
```

The `reason` variable holds the reason for triggering the method, in this case, was it a left click or a right click. The left click and the right click code was implemented in the `__init__` method at the beginning of the class definition. The right click code pops up a menu with two items, 'About' and 'Exit', which are in turn connected to methods that perform the selected function. The left click code shows the form and connects the two buttons to the relative method.

The final part is this:

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    app.setQuitOnLastWindowClosed(False)
    form = Pad()
    app.exec_()
```

The highlighted line is the only change here. It tells the application not to close when one of its windows are closed.

If this looks like a lot of code, then remember that this is **not** python code. This is PyQt code, which allows python to connect with the Qt system. The only python code here is what you see with all of the methods closed (OK, the file handling routine is python stuff), and then you have little more than we started with when writing the template code (right).

True python code is the sort we used in the application finder application, slicing and splitting strings and rebuilding them, creating lists and lists of lists then looping through them.

Gradually I will include more of that in these examples. With these examples, you collect a fair amount of PyQt code that can be copied into your own applications, and re-used with little or no modification.



Like Us On Facebook!
The PCLinuxOS Magazine
PCLinuxOS Fan Club



```
scratchpad.py x qt5_template.py x
1  #!/usr/bin/env python3
2
3  import sys
4  from PyQt5.QtCore import *
5  from PyQt5.QtWidgets import *
6  import template_ui
7
8
9  class Template(QWidget, template_ui.Ui_Form):
10     def __init__(self):
16
17     def keyPressEvent(self, e):
22
23     def clearText(self):
26
27     def exitApplication(self):
30
31
32  if __name__ == '__main__':
```

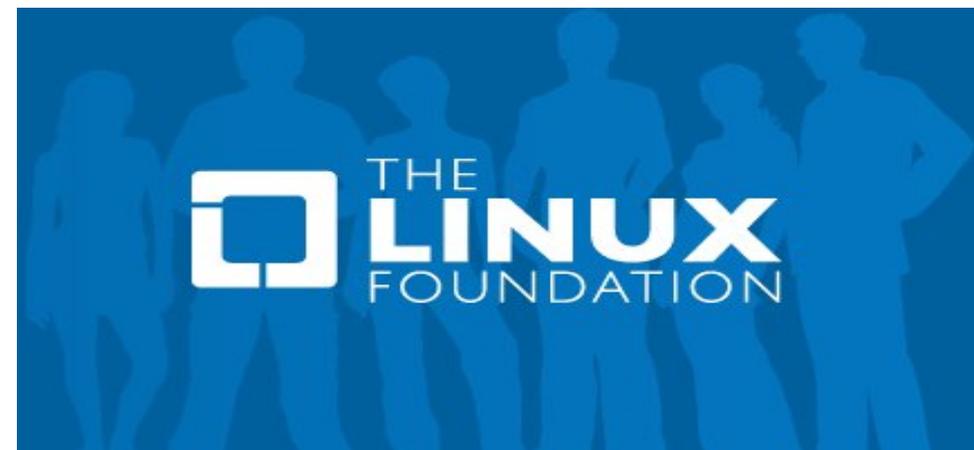


PCLOS-Talk

Instant Messaging Server

Sign up TODAY! <http://pclostalk.pclosusers.com>





Disclaimer

1. All the contents of The PCLinuxOS Magazine are only for general information and/or use. Such contents do not constitute advice and should not be relied upon in making (or refraining from making) any decision. Any specific advice or replies to queries in any part of the magazine is/are the person opinion of such experts/consultants/persons and are not subscribed to by The PCLinuxOS Magazine.
2. The information in The PCLinuxOS Magazine is provided on an "AS IS" basis, and all warranties, expressed or implied of any kind, regarding any matter pertaining to any information, advice or replies are disclaimed and excluded.
3. The PCLinuxOS Magazine and its associates shall not be liable, at any time, for damages (including, but not limited to, without limitation, damages of any kind) arising in contract, tort or otherwise, from the use of or inability to use the magazine, or any of its contents, or from any action taken (or refrained from being taken) as a result of using the magazine or any such contents or for any failure of performance, error, omission, interruption, deletion, defect, delay in operation or transmission, computer virus, communications line failure, theft or destruction or unauthorized access to, alteration of, or use of information contained on the magazine.
4. No representations, warranties or guarantees whatsoever are made as to the accuracy, adequacy, reliability, completeness, suitability, or applicability of the information to a particular situation. All trademarks are the property of their respective owners.
5. Certain links on the magazine lead to resources located on servers maintained by third parties over whom The PCLinuxOS Magazine has no control or connection, business or otherwise. These sites are external to The PCLinuxOS Magazine and by visiting these, you are doing so of your own accord and assume all responsibility and liability for such action.

Material Submitted by Users

A majority of sections in the magazine contain materials submitted by users. The PCLinuxOS Magazine accepts no responsibility for the content, accuracy, conformity to applicable laws of such material.

Entire Agreement

These terms constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes and replaces all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter.

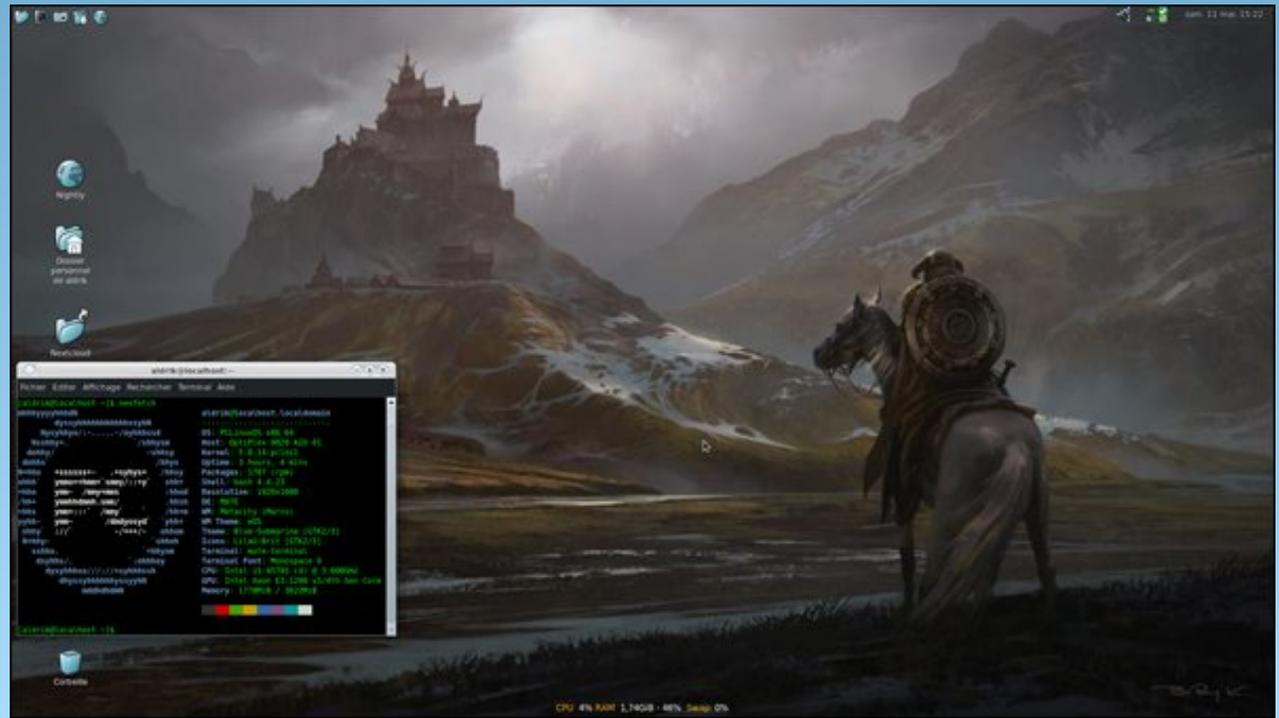


CHIMPBOX

Big Things, Little Packages. Measuring only 7.5 X 8.5 X 2 inches

<http://chimpbox.us>

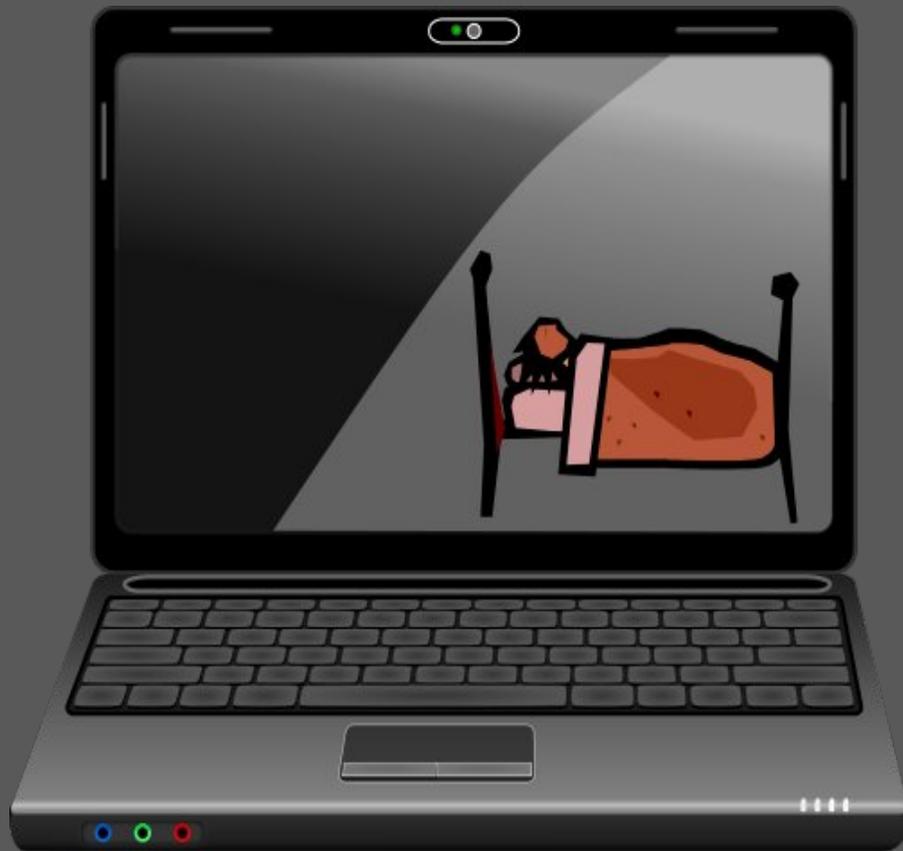
Screenshot Showcase



Posted by Aldrik, May 11, 2019, running Mate.



ms_meme's Nook: Black Painted Desktop



MP3

OGG

I dream of my desktop it is painted black
Nightmare of Windows or might it be a Mac
Ordeal of horror image of woes
Mind tripping agony you know how it goes

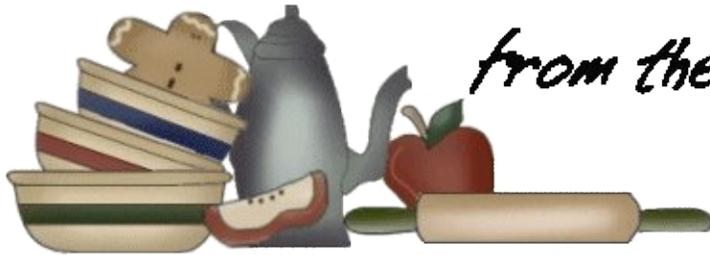
I dream of my desktop bottomless pit of black
Haunting illusion think I'm going to crack
Turning my eyes I hope it goes away
Trying to forget I used Windows every day

I look into my 'puter has it been hacked
Is it possible my files have been hijacked
Ghastly and gloomy dark dank and grim
Foreboding baneful outcome looking dim

I dream of my desktop it is painted black
Nightmare of Windows or might it be a Mac
Ordeal of horror image of woes
Mind tripping agony you know how it goes

Rousing from my drowsing it is now blue
A Linux desktop a wonder through and through
Aura of beauty what a success
Happiness for me I am using PCLOS

PCLinuxOS Recipe Corner



*from the kitchen of
youcantoo*

Crispy Whole Chicken and Vegetables

Ingredients:

1 whole chicken (3 1/2 lb)
3 tablespoons olive oil
2 tablespoons Montreal chicken seasoning
1 1/2 lb baby (B-size) red potatoes, quartered
1 medium red onion, cut into wedges
1 tablespoon chopped fresh thyme leaves
3/4 teaspoon salt
1/4 teaspoon pepper

Directions:

1. Heat oven to 400F. Place chicken, breast side down, on cutting board. Using heavy-duty kitchen scissors or poultry shears, cut closely along one side of backbone from thigh end to neck. Repeat on other side. Remove backbone, and discard. Turn chicken over so breast side is up; press down to flatten breast area by pressing firmly with heel of hand. Rub chicken with 1 tablespoon of the oil.

2. Heat 12-inch cast-iron skillet over medium-high heat until hot; place chicken breast side down. Season side facing up with 1 tablespoon chicken seasoning; top with 10-inch heavy skillet to weigh down. Cook 5 to 7 minutes or until skin is golden brown. Remove smaller skillet, and transfer chicken to plate.

3. In medium bowl, toss potatoes, onion, remaining 2 tablespoons oil, the thyme, salt and pepper until mixed well. Arrange vegetable mixture in single layer in the 12-inch cast-iron skillet. Place chicken, breast side up, on vegetables, and sprinkle top with remaining 1 tablespoon chicken seasoning.

4. Roast uncovered 45 to 55 minutes or until juice of chicken is clear when thickest pieces are cut to bone (at least 165F), and vegetables are tender and lightly browned. Garnish with additional thyme leaves, if desired.

Use caution when removing skillet from oven by using two heavy ovenproof mitts. The skillet (and handle) will be very hot.

Be sure to buy a chicken that's between 3 and 3 1/2 lb. Anything larger may not fit inside the skillet.

Expert Tips

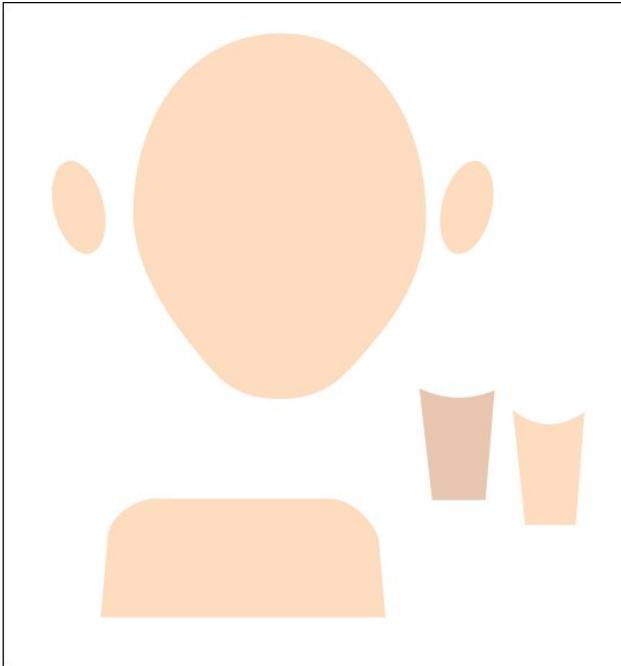
Cast-iron skillets are great because of their heavy weight and their ability to retain heat. If you don't own two heavy skillets, try a weighted smaller skillet, or wrap bricks in foil, and use in place of a skillet.



Inkscape Tutorial: Create An Avatar

by Meemaw

An avatar is the graphical representation of someone, often used in social networks, forums, games etc. Usually, a 2D-avatar is a square image which has a small size (100x100px, 64x64px). I'm going to create an avatar for myself, and you can do the same. We'll start with simple objects like circles, ellipses, rectangles and squares.



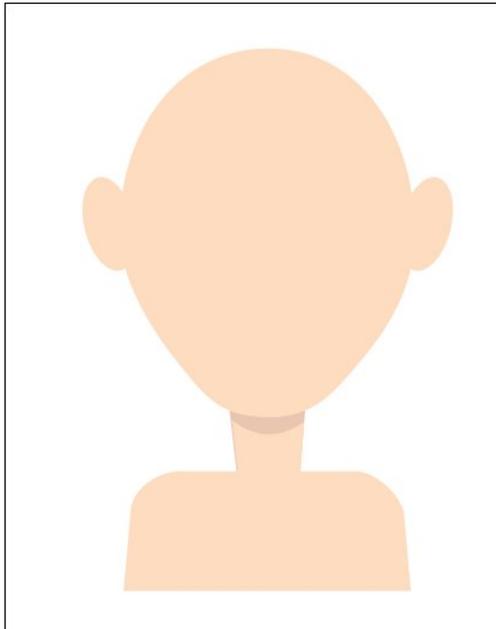
Create a circle using the circle tool. Convert the circle to path using **Path > Object to Path**. Edit the path by the nodes to make it look like a human head.

Create an ellipse and edit its nodes, so that looks like an ear. Duplicate it (**Ctrl+D**) and flip it horizontally. Place both ears in the middle of the head.

To create the neck, draw a rectangle, convert it to a path and place two lower nodes closer to each other. An easy way is to select them and press **Ctrl+<** repeatedly.

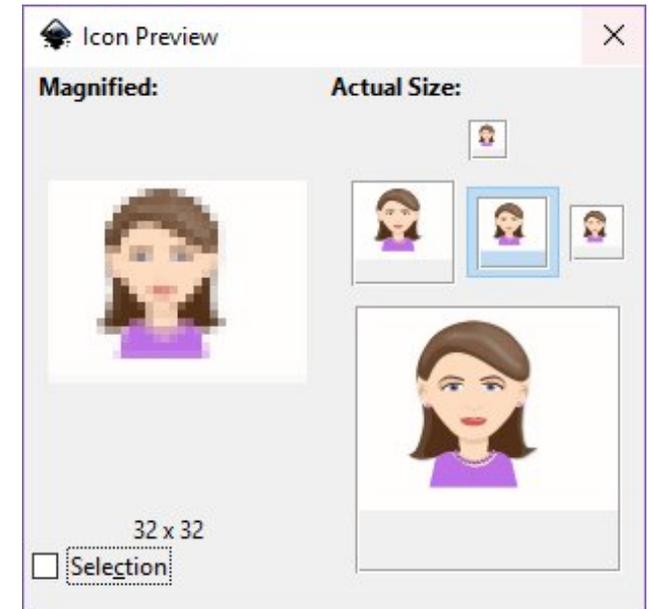
Create shoulders with the rectangle tool and make the top nodes smoother. Make the shoulders narrower than normal to focus the viewer's attention on the head.

In our shape collection above, we see that one is darker than the rest... that is a duplicate I made of the neck. Using that duplicate along with the original, you can create a "separation" between the head and neck in the form of a shadow.



There are a couple of useful features in Inkscape that can help you create avatar easily:

View > Icon Preview will let you preview the page or selected objects as an icon in different resolutions (16x16px/24x24px/32x32px/48x48px and 128x128px). Here's what I get:

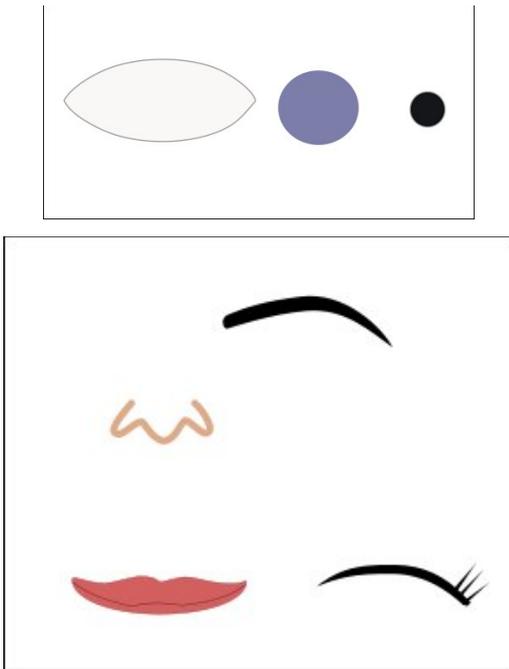


It shows the icons for several sizes on the right, and will show on the left what you get if you try to make your icon any bigger. It will show you how your creation looks in icon size, to help you determine how you want to proceed, as far as detail and how much to include. For example, you could zoom in and concentrate on the face and omit the shoulders.

View > Duplicate Window will open a new window with the same document in which you are working. Editing the original document will duplicate the changes in the second one. However, you can make

changes in both windows. This feature can be useful when working in Inkscape as a whole, not just for this project.

Next, work on the facial features..



Usually eyes have an almond shape and are at the same level with the top edges of the ears. Create an ellipse and edit the nodes to get it the shape you want. Create two circles for iris and pupil.

Draw an eyebrow with Bezier curve in “Triangle in” shape mode. Convert the object to path and smooth needed nodes.

Draw a lash line with Bezier curve in “Ellipse” shape mode. Separate lashes can be easily done with Bezier curve in “Triangle in” shape mode. You can arrange them the way you want and then group them.

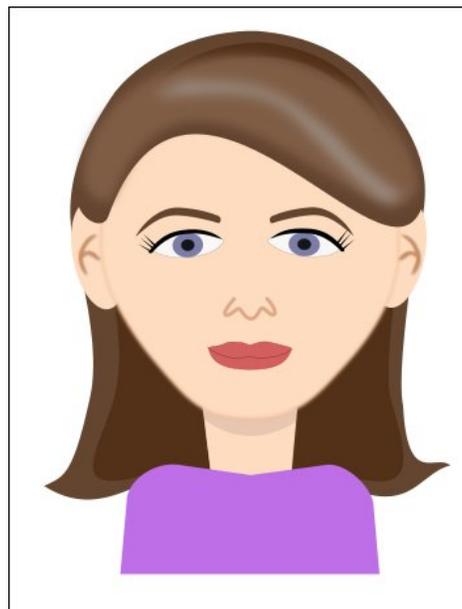
Add details to the ears by using Bezier curve in “Ellipse” shape mode. Convert both objects to path

and give them the desired shape, then group them. If there are a large number of unnecessary nodes, go to **Path > Simplify** or just press **Ctrl+L** after selecting the path.

Also, many head elements are symmetrical and we can make things much easier. Simply create only one eye, duplicate it (**Ctrl+D**) and flip horizontally.

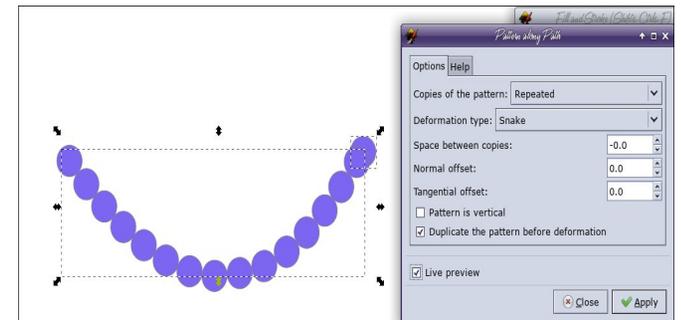
You can also use your Bezier tool to shape some hair for yourself, and edit the nodes to shape it the way you want. I made a couple of shapes in blending colors, because the light gives everyone’s hair different highlights (and I also have a big gray streak in the front). Stack them however they look correct. This will also give depth & some shading to your avatar.

Now apply them to your avatar in the places desired. Mine is pretty simplistic, but it might be close to what I look like. Note: I’m terrible with facial features! However, just because I made my avatar similar to myself doesn’t mean you have to. You can make yours look like anything or anyone ... it’s your creation!



Inkscape Tutorial: Create An Avatar

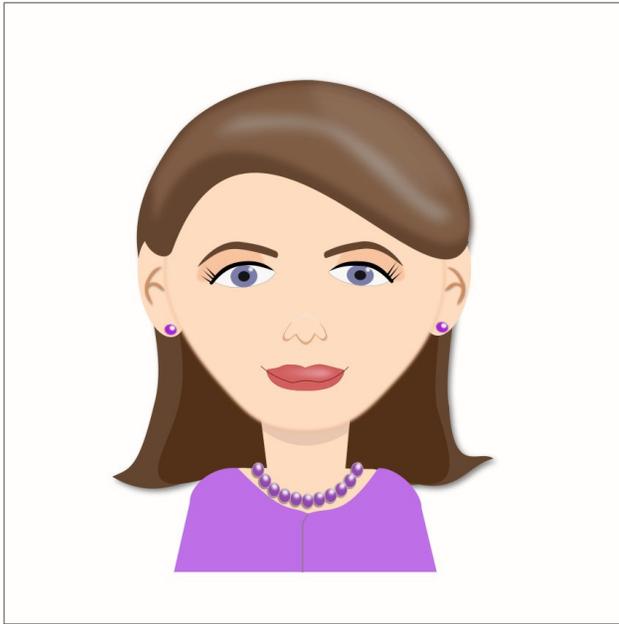
In the tutorial I read, the next step was to add color, but I’ve been coloring everything as I went along. The next step here is to add some extra detail, like highlights where needed, and even jewelry if you want it. I added some eye shadow (I wear a kind of golden beige, so it really doesn’t show all that much), highlights in my eyes and on my lips and hair, and edited the neckline of my shirt to add a necklace. It’s easy! We’ve used this extension before. Create one bead with the Ellipse tool, then draw a necklace path with the Bezier tool. Use the Pattern along Path extension by clicking **Extensions > Generate from Path > Pattern along Path**. Make sure that you place the bead on top, otherwise the extension won’t work correctly. To see the result before you apply, make sure you click Live Preview. You can make the highlights the same way. (Make earrings to match!)



You should get one object consisting of many beads along the path.

All that’s left is some fine tuning, like shadows on the face. Add more details to the hair — most of this can be done with lines using the Bezier tool. After that, add shadows under the eyes, a seam/opening on the shirt and pink highlights on the lips if you haven’t already done it. How involved you want to get is entirely up to you. Now export your avatar! You can export an icon sized avatar as well. This one is 128 x 128.





Screenshot Showcase



Posted by francesco bat, May 5, 2019, running IceWM.

Does your computer run slow?

Are you tired of all the "Blue Screens of Death" computer crashes?



Are viruses, adware, malware & spyware slowing you down?

Get your PC back to good health TODAY!

Get



Download your copy today! FREE!



De-Googling Yourself, Part 2



by **Alessandro Ebersol (Agent Smith)**

Google is a threat to the privacy and the very daily life of its users. Why? Because Google profiles its users. And, how did it all begin?

Profiling to better serve ... (?)

To get better search results, Google started profiling its users. And thanks to that, the search results have been improving over the years.

But Google needed a better way to capture users' personal information, to better work the search results. And, what better way than the users themselves giving their personal data to Google?

So, in 2004, Gmail was born. This was the beginning of the Google Office suite (G-suite), initially a service that only accepted users by invitation (in a period when it was labeled as a beta program). It was opened to the general public in 2009, five years later.

And so, Google began to profile the users of its search service through its email service.

To get an idea of how profiling works, try searching Google when you're logged in your account, and then search the same terms when you're logged out of your account to see how the results look different.



How Gmail reads emails (and, time to freak out now)



Google's email servers automatically check email for a variety of purposes, including adding contextual ads alongside emails and spam and malware filtering. This is the official version of why your messages are read. But the innocent and noble causes can lead to much darker consequences.

Privacy advocates have raised many concerns about this practice. The concerns include that since email content is read by a machine (as opposed to a person), it may allow Google to keep unlimited amounts of information forever. Automated background scanning increases the risk that the expectation of privacy in the use of e-mail will be reduced or eroded. The information collected from emails can be retained by Google for years after its current relevance to create complete user profiles. Emails sent by users of other email providers are verified, although they have never agreed to Google's privacy policy or terms of service. Google may change its privacy policy unilaterally, and for minor changes to the policy, it may do so without informing users. In court cases, governments and organizations may find it easier to monitor e-mail communications legally. At any time, Google may change the company's current policies to allow the combination of e-mail information with data collected from the use of other services. And, any internal

security issues on Google systems can expose many - or all - of their users.

So much so that in January 2010, Google detected a "highly sophisticated" cyber attack on its infrastructure originating in China. The targets of the attack were Chinese human rights activists, but Google found that accounts owned by European, American and Chinese human rights activists in China were "routinely accessed by third parties." Thus, data collected by Google has been abused by third parties.

Google, on June 23, 2017, announced that by the end of 2017 it would phase out the scanning of email content to generate contextual advertising, relying on **personal data collected through its other services**. The company said that this amendment was intended to clarify its practices and lessen the concerns of G Suite business customers, who felt an ambiguous distinction between the free consumer and the paid professional variants, the latter being free of publicity.

However, the above highlight demonstrates that, rather than diminishing, Google's meddling has increased. This is because of all the products Google offers, and its ability to monitor the use of those services.

Google creates many tools that are useful to users and administrators, but they are designed to collect user data and create as much of a profile as possible, which is sold to advertisers, governments, other data brokers, or anyone else who wants to pay. The difference is that the method Google uses is aggregation.

I'll list some of Google's popular (and some even transparent to the end user) services and products that profile their users, and help the company catalog the online habits and actions of those who use them.



Google AMP (Accelerated Mobile Pages)



Google AMP is a service that stores data, usually media, on Google servers around the world. This means that when you upload a website with AMP enabled, the images and media come from Google's servers. This means that when you visit a website with AMP enabled, Google knows all the features you've loaded on the page. Interestingly, this gives Google access to substantially more information than your internet provider could get, because HTTPS encryption prevents the provider from seeing the specific pages you visit. They can only see the domain. For example, your ISP might see that you visited Reddit, but not what you visited on Reddit. The linked Google AMP content on Reddit (and there's a ton of it) gives Google a direct IP link, content that they can document and use to map user behavior and activity profiling.

This problem is widespread. WordPress sites, which are the most popular content management system in the world, have AMP enabled by default.

Worse, Google has recently announced that Chrome mobile users will not even know when they're using amp-served content. Chrome will hide the AMP content behind the original URL.



Google Analytics



Google Analytics uses cookies and cross-site tracking to identify and track users as they browse the Web in their daily routines. Google Analytics works by assigning a user a cookie with a unique ID number, and every time a user visits a site with Google Analytics enabled, Google records that activity and links it to the user with that particular cookie. Many times, this is done without the user being aware.

Recently, the European General Data Protection Regulation (GDPR) has created many warnings to users about this type of analytical software, but has been inefficient in restricting its use because Google Analytics is so ubiquitous that it bothers users with cookie warnings at almost all sites visited. Many do not follow GDPR's rules and allow users to browse the site or use services without tracking.

Google Cloud



Another great data point for Google is the Cloud. As of 2018, Google hosts about 9.5% of all cloud content in the Google Cloud (by revenue, most Google cloud services are "free" and they can host much more by volume). If you're using an application

or website that uses the Google Cloud infrastructure, this is another gateway for your information. A user does not grant Google permission to retain data about them to use Google Cloud services.

Google Maps API



Every time you visit a company website and use Google Maps (not a screenshot), it uses the Google Maps API. This data is combined with Google Analytics to attach location information to your Google profile.

Google FireBase



FireBase is a tool that allows developers to easily synchronize data between different sites, applications and services. The caveat is that this data is synchronized through Google's servers, which record all this data and profile without the user's knowledge.

Google Chrome



Google Chrome records everything you've searched for on Google Search, all the websites you've visited or tagged, all the YouTube videos you've watched, all the ads you've clicked on, and how many passwords were automatically filled by Google Chrome.

These registered browser habits of Google Chrome include:

- * Everything you've searched using Google Search or YouTube
- * Your YouTube history
- * How many Google searches you've done during this month
- * All the sites you've already clicked on
- * Every website address that you have inserted in the address bar
- * All the sites that you have already checked
- * All Google Chrome tabs that are open on all your devices.
- * How many Gmail conversations you've had
- * The apps you've downloaded from the Chrome Web Store and Google Play store
- * Your Chrome Web Store extensions

- * Your Google Chrome browser settings
- * All email addresses, street addresses, phone numbers you've set to fill automatically on Chrome
- * All the usernames and passwords you've asked Chrome to save
- * All the sites you've asked Chrome to not save passwords

And, by the end of 2018, Google Chrome automatically logs you in when you access Google sites. That is, almost mandatorily, the user is forced to share his/her activity online with Google.

Android devices



We come to a very important point, and that is also perhaps the most vulnerable point of the user's relationship with Google, since the Android operating system, the Google application store (Google Play) and most (if not all) of the services of Google (GSF: Google Service Framework) are an inexhaustible source of enterprise intrusion into the personal lives of its users.

To download and use Google Play Store apps on an Android device, you must have (or create) a Google Account, which becomes an important gateway through which Google collects personal information including user name, email, and phone number. If a user signs up for services like Google Pay, Android will also collect credit card information, zip code and the user's date of birth. All of this information



The PCLinuxOS Magazine

Created with Scribus

becomes part of an user's personal information associated with his/her Google Account.

In addition to personal data, Chrome and Android send information to Google about browsing activities and mobile apps, respectively. Any visit to a webpage is tracked and automatically collected under Google's user credentials. If you sign in to Chrome, the browser will also collect information about browsing history, passwords, site-specific permissions, cookies, download history, and additional user data.

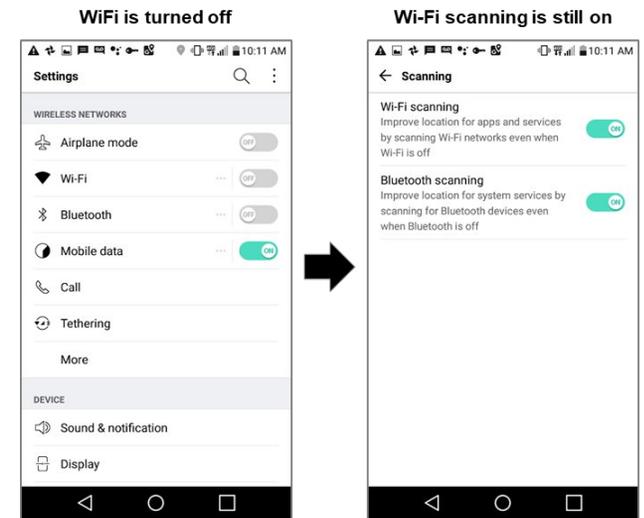
Android sends periodic updates to Google's servers, including device type, mobile service provider name, crash reports, and information about apps installed on the phone. It also notifies Google whenever an application is accessed on the phone (for example, Google knows when an Android user accesses the Uber application).

The Android and Chrome platforms meticulously collect user location and motion information using a variety of sources, as shown in the figure below.

For example, a "rough location" rating can be done using GPS coordinates on an Android phone or via a laptop's network IP address. User location accuracy can be further improved ("good location") by using nearby cell tower IDs or by scanning device-specific BSSIDs or basic service set identifiers assigned to the radio chipset used in access points near WiFi hotspots. Android phones can also use information from registered Bluetooth beacons with the Google Proximity Beacon API. These beacons not only provide the geolocation coordinates of the user, but can also identify the exact floor levels in buildings.

It is difficult for an Android mobile user to "refuse" location tracking. For example, on an Android device, even if an user turns off WiFi, the location of the device will still be monitored through WiFi. To avoid this tracking, WiFi scanning must be explicitly disabled in a separate action as shown on the right.

The omnipresence of WiFi hubs has made location tracking quite frequent.



For example, in a study conducted by Professor Douglas C. Schmidt, of Vanderbilt University, during a short 15-minute walk around a residential neighborhood, an Android device sends location requests to Google. The request contained collectively about 100 unique BSSIDs from public and private WiFi access points.

Google can check with a high degree of confidence whether a user is standing, walking, running, cycling or riding a train or car. It achieves this by tracking the location coordinates of an Android mobile user at frequent time intervals in combination with panel sensor data (such as an accelerometer) on mobile phones.

But it does not stop there. Because of the infrastructure that Google services enable, there is a whole host of enterprise products that are used daily by advertisers and internet advertising companies.



Google products for online ads and advertisers

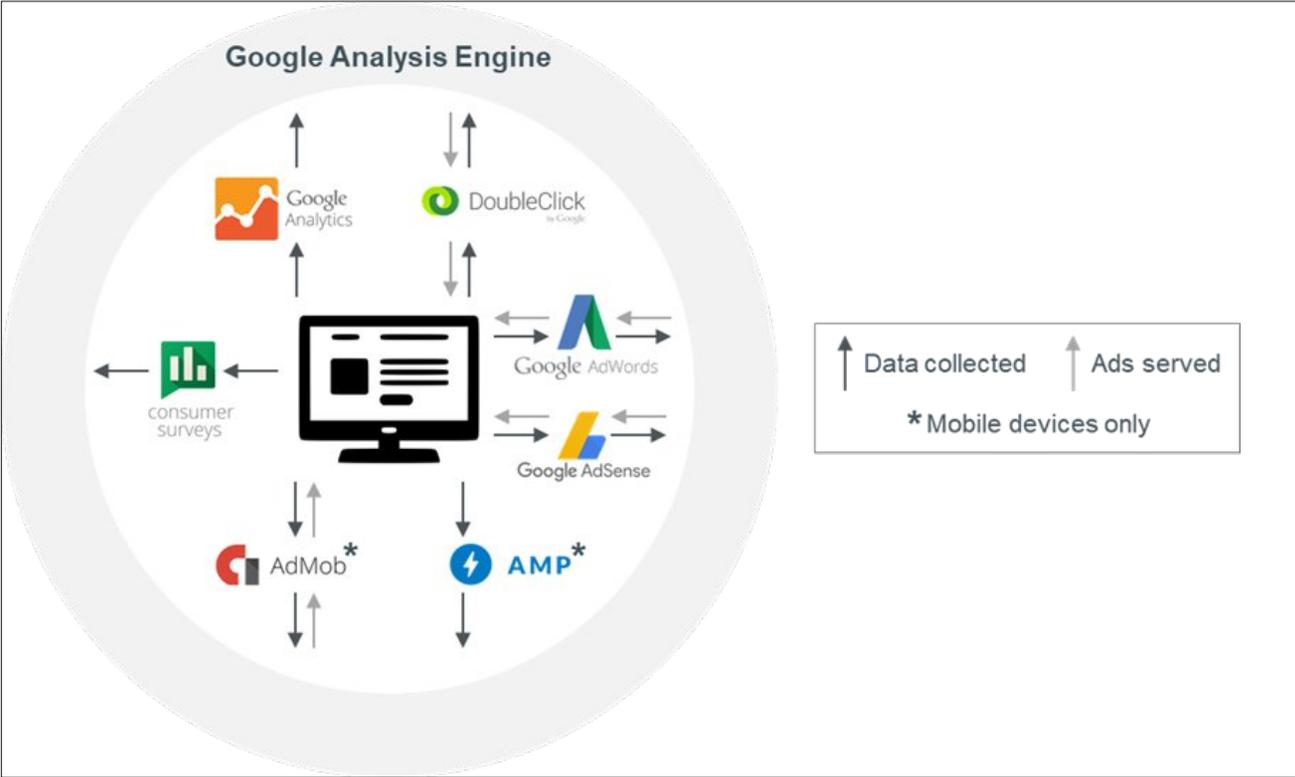


An important source for Google user activity data collection mechanisms are its publisher and advertiser tools such as Google Analytics, DoubleClick, AdSense, AdWords, and AdMob. These tools have a huge reach, for example. more than a million mobile apps use AdMob, over 1 million advertisers use Google AdWords, more than 15 million websites use Google AdSense, and more than 30 million websites use Google Analytics.

There are two main groups of users of the Google tools that are focused on publishers and advertisers:

* Editors of sites and applications, which are organizations that have websites and create applications for mobile devices. These entities use Google's tools to make money by allowing ads to appear on their websites or applications, and to improve tracking and understanding who is visiting their websites and using their applications. Google's tools place cookies and run scripts on website visitors' browsers that help determine a user's identity, track their interest in the content, and track their behavior online. Google's mobile application libraries track application usage on mobile devices.

* Advertisers, which are organizations that pay to display banners, videos, or other ads to users while browsing Internet or, use applications. These entities apply Google's tools to target specific people profiles in ads to increase the return on their marketing spend (better targeted ads generally lead to higher click-throughs and conversions). These tools also allow advertisers to analyze their audiences and evaluate the effectiveness of their digital advertising



by tracking which ads were clicked on that frequency and providing information about the profiles of the people who clicked the ads.

Together, these tools collect information about user activity on websites and applications, such as visited content and clicked ads. They work in the background - largely imperceptible to users. The figure below shows some of these main tools, with arrows indicating data collected from users and ads displayed to users.

Conclusions

It's frightening how much intrusion Google has in our lives, whether we know it, or worse, without knowing anything. Without suspicion, we live in a birdcage, where advertisers, big corporations and

governments know of all our steps, our tastes, our political and religious orientation.

Our secrets are no longer ours, but, they are with the corporations now: Google, Apple and Facebook know when a woman visits an abortion clinic, even if she does not tell anyone else. The GPS coordinates on the phone do not lie. Extramarital affairs are an easy thing to imagine: two smartphones that have never met before, cross into a bar and then head to an apartment across the city, stay together at night and leave in the morning ... well, you can imagine the rest.

It's a sad brave new world. But not everything is lost. It is possible to improve privacy and lead a more reserved life. We will see how, in the next article. So, stay with us, next month for more.



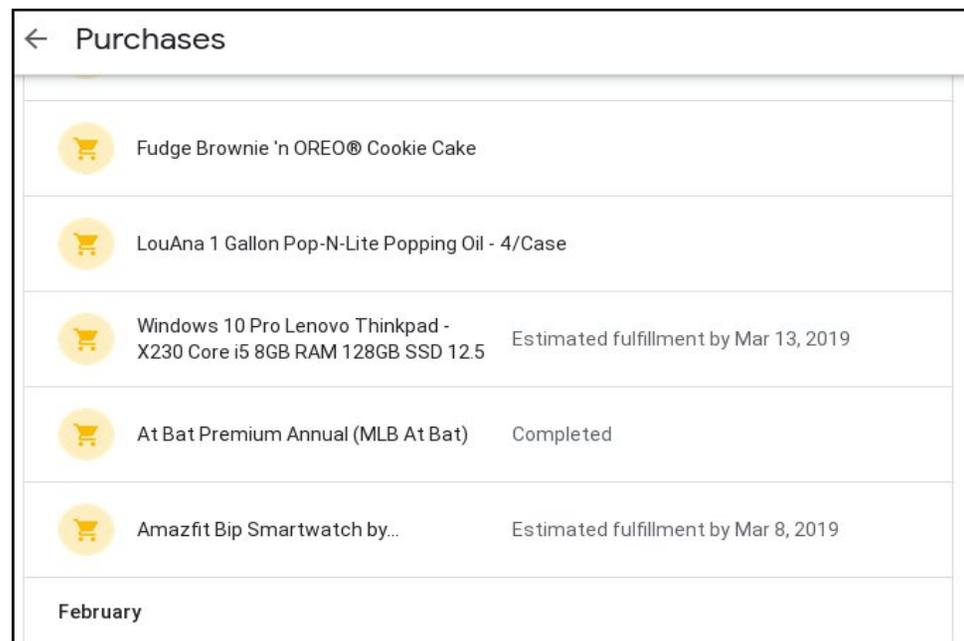
Gmail Eerily Tracks Your Purchases

by Paul Arnote (parnote)

Now just when or from where did you purchase those multi-colored shoe laces? If you purchased them online, or if you had a copy of your receipt emailed to your Gmail account, Google knows. And, Google remembers.

Unceremoniously and quite stealthily, Google has amassed a wealth of information about your purchasing history. Every item you've ever purchased online (and used your Gmail email address for purchase confirmation) and every time you've had a merchant you do business with email you a copy of your receipt to your Gmail email address, that information has been extracted and stored by Google. This shocking revelation came out in a May 17, 2019 [article](#) on CNBC.

Here's one scary part about all of this: your purchase history is likely to go back YEARS. My own purchase history goes all the way back to 2013. Some people, like Nick Statt on [The Verge](#), have their purchase history go all the way back to 2010.

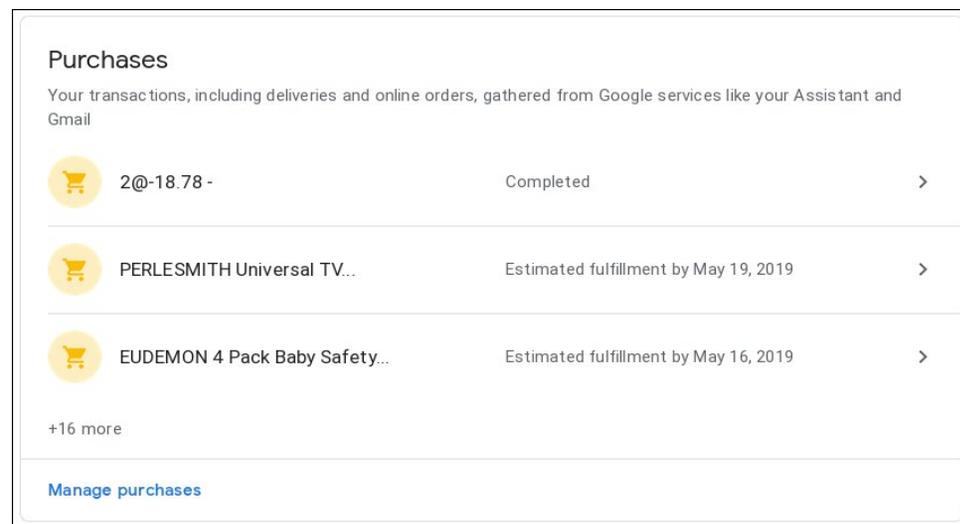


The small snippet shown in the screenshot above is from March, 2019 of my own purchase history that Google has saved. Until I ran across the CNBC article, I never even knew that Google was keeping track of my purchases. That tends to be the surprise reaction from most people.

So, let's take a look at what the information in the small snippet above shows, working from the bottom of the image to the top. Well, there is the Amazfit Bip Smartwatch I bought for my wife for her birthday. Then there is my annual subscription renewal to my MLB At Bat account. In the early part of March, I purchased a new small form factor laptop to replace my out-of-date and defunct netbook. I did purchase some popcorn popping oil for the popcorn popper we have at work from a restaurant supply company. Oh, and there is the ice cream cake from Baskin Robbins that I ordered and purchased online for my wife's birthday.

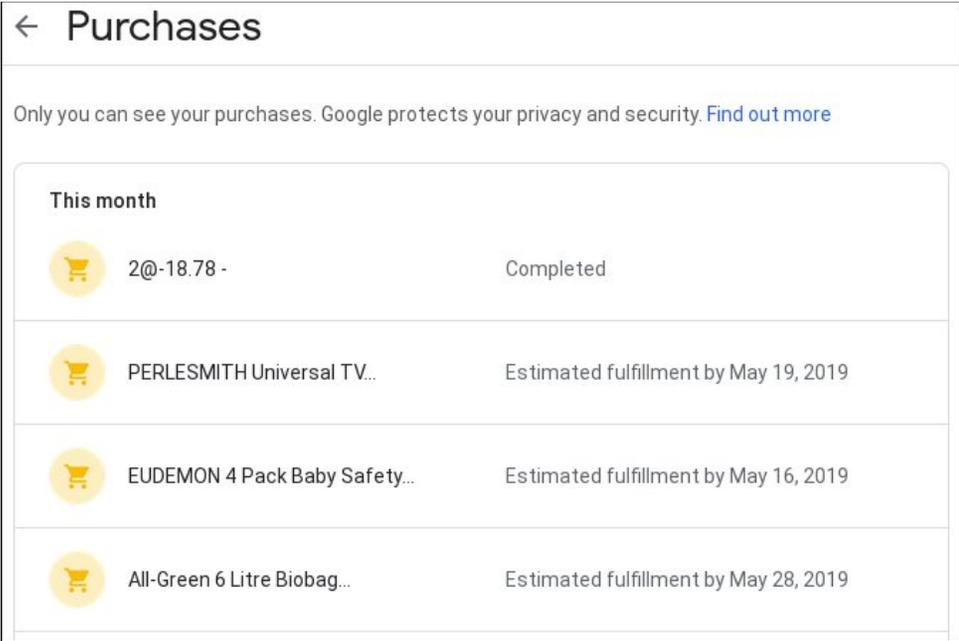
So how revealing is this information? Well, it shows that my wife most likely has a smartphone, since most smartwatches connect to smartphones. It shows that we are more than a little bit tech savvy. It shows that I happen to like baseball and popcorn. Oh, and it shows that we (my wife and me, at least) are chocoholics. And that would be just the tip of the iceberg. The information you can infer from all of this is massive.

Don't worry ... this gets better. Follow along here.

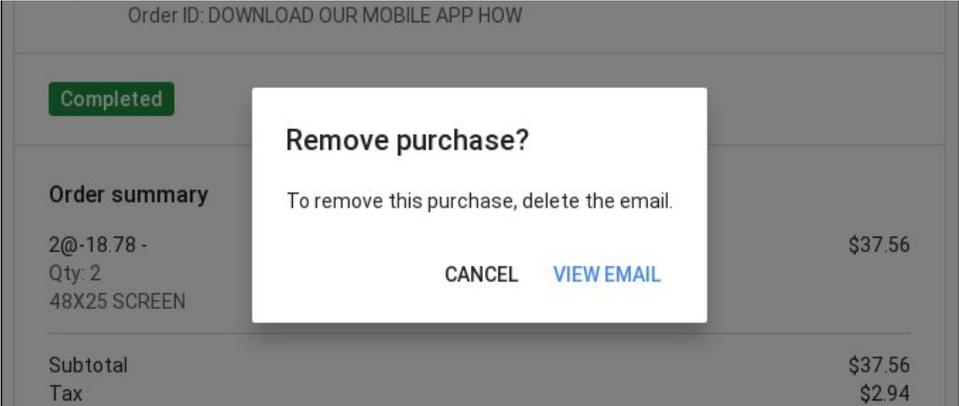


Gmail Eerily Tracks Your Purchases

Go [here](#), and you can view all of your payments and subscriptions that Google tracks. Just below the section where Google tracks your payment methods, you will find the “Purchases” section. Click on the “Manage purchases” link at the bottom of the box.



Notice how, at the top of the resulting screen, Google informs you that “only you can see your purchases.” Yeah, right. Only me ... AND Google! So, I clicked on the first item in the list of purchases, which wasn't a purchase at all. It was a return. I returned some wrong size screen material to Home Depot that I had purchased in error. I just had the receipt sent to my Gmail account.



Clicking on the “purchase” brings up a window that displays all the sorted details of my transaction. Then, clicking on the “Remove Purchase” link at the bottom of the transaction results in the display of the new popup window. Seriously? To remove the purchase, I have to delete the email? This shouldn't be my only option. Maybe I need to save the receipt that is included in my email for future reference or warranty claims.

From the CNBC article:

“To help you easily view and keep track of your purchases, bookings and subscriptions in one place, we’ve created a private destination that can only be seen by you,” a Google spokesperson told CNBC. “You can delete this information at any time. We don’t use any information from your Gmail messages to serve you ads, and that includes the email receipts and confirmations shown on the Purchase page.”

Except, there is no easy way of deleting your purchase history unless you delete the email containing the receipt for the transaction. Ok. Let's see if there's a way to do this from another angle. Let's go to Google's activity control [page](#). Certainly, there must be some setting I can make there to prevent Google from tracking my purchases.

Nope. There is NOTHING there to manage whether Google tracks your purchases, either. This is the other scary aspect of Google tracking your purchases. There is NO WAY, short of deleting the email containing the information, to turn off purchase tracking or deleting information from your purchase history.

Google vehemently asserts that nothing in or from your Gmail account is used to target ads to you (don't forget that Google's primary source of income is from serving you ads). In fact, the contents of your Gmail account has not been used to serve you ads since they stopped the practice in 2017. They also claim to not sell your data to **advertisers**. So, what about **other** entities? Let your imagine roam on just who those “other entities” could or might be. You probably won't land too far away from the target bullseye.

So, if Google isn't using the information to target ads to serve up to you, then why are years of purchase information being stored? Most people don't even know that Google is doing this. Also, if Google isn't using this information, then what is the purpose of saving years of purchase information? And, why make it so hard to delete that information?

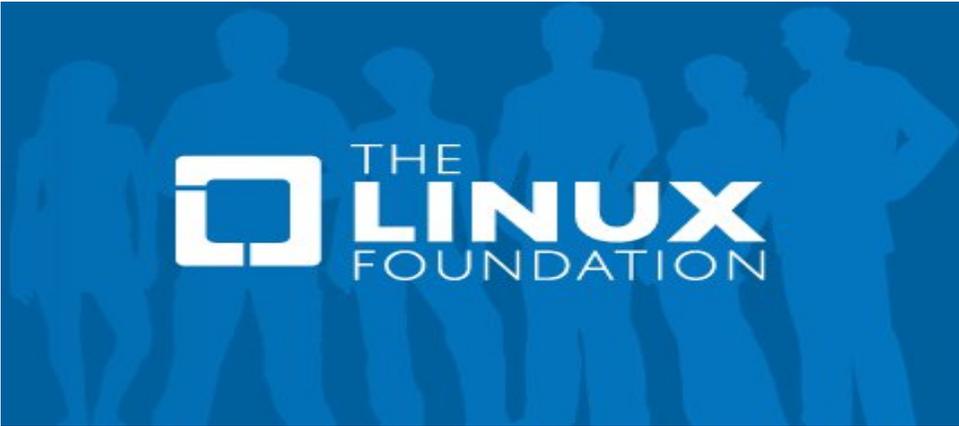
Google has said that it is looking into making the settings simpler and easier to control. Still, Google has obviously abandoned “Do No Evil” as a corporate motto. Instead, it seems more like the programming for V-Ger in Star Trek: The Motion Picture. Collect all data that can be collected.



Fortunately there may be one way to thwart Google's collection of your purchase history. The ONLY reason my purchase history only goes back to 2013 (despite having a Gmail account from back in the beta days in 2005 or before) is that I **used** to use an email program to download my emails from Gmail, and then delete them from Gmail after they were downloaded to my computer. That is the ONLY rational explanation I can conceive of for why my purchase history doesn't go back any further than 2013, while other users have purchase history going back even farther in time. Since at least 2013, I have exclusively used the Gmail webmail interface for all of my Gmail interactions.



Google's behavior reminds me of a popular internet meme. Obviously, no one at Google has ever heard of the maxim "just because you can, doesn't mean that you should." If they have, then the whole meaning of it has become lost on them.



The PCLinuxOS Magazine Special Editions!

Get Your Free Copies Today!





Support PCLinuxOS! Get Your Official

PCLinuxOS
Merchandise Today!

PCLinuxOS



Help PCLinuxOS Thrive & Survive

DONATE TODAY



Screenshot Showcase



Posted by mutse, May 3, 2019, running Mate.

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

by phorneker

The past two articles I have written on this subject presented us with the basic concepts of programming in Ruby, namely the structure and execution flow of a Ruby program, along with the basic elements such as variables and input/output statements. Ruby was designed to be an object oriented language. However, we have seen that it can be used for structured programming. This means:

1. Programs have only one entry point and one exit point,
2. Functions and procedures are implemented for often used tasks within programs.
3. All variables used **are declared at the beginning of the program**.
4. When programs run properly, the code is easier for people to read and understand.

Object oriented programming, on the other hand:

1. Encapsulates data and program code into one abstract package called a **class**.
2. Places more emphasis on expected behaviors of each **class** than on the structure of its data and functions.
3. Allows reuse of variable names, functions, operators (such as +, -, >, and <) and even entire classes.

With object oriented programming, **everything is an object**, as we have seen so far in Ruby.

One of the reasons why I prefer **structured programming** to **object oriented programming** is that when programs are designed with structured programming, it is much easier to follow the flow of execution, ensuring that the functions and procedures are **executed correctly** and are much easier to debug.

This is more often not the case with object oriented programming, where

the flow of execution depends on how the class used by the object was implemented. This makes it not so easy to identify and debug errors.

In this article, we will discuss how Ruby handles variables, classes, arrays and strings.

What We Know About Variables

Variables are the basic elements of data storage in any program. In Ruby, variables themselves are not assigned any value, nor are they of any data type. They are simply a pointer to locations in memory assigned for data storage when the program is executed.

For a variable to be useful, it must be assigned a value, be it a direct assignment, or in a control loop such as a **for** loop. When a direct assignment is made to a variable, the value assigned to the variable determines the data type for the variable.

So far, we know that a variable can be of **floating point**, **integer**, or **string** as far as what we call data types that can be assigned to a variable.

Types of Variables

In addition to data types, Ruby offers five **types of variables** that can be incorporated into programs.

In the last article, I mentioned that program code can be encapsulated into blocks. These blocks are used within control statements such as **for**, **while**, and **unless** between the statement and the **end** that defines the end of the block of code.

Local variables are variables that are defined **within** these blocks of code and can be accessed **only from within that block of code**.

Defining variables within a block of code ensures that data needed within such a block of code, such as iterators within a loop are contained **within that block**, and not accessible from outside the block.

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

This is an example of what we call **information hiding** and is a good programming practice.

In C and C++, the **private** keyword makes variables accessible only to the function for which the variable is defined. The **protected** keyword, however, allows descendants of that function to access the variable, which would otherwise be inaccessible.

Global variables are variables that can be accessed from **anywhere** in the program.

Note: A variable declared outside of any function or block of code is considered to be **global** *within that particular source code file*.

As a good programming practice in Ruby, global variables should be prefixed with a "\$". It is also a good practice to keep the number of global variables to a minimum, making the program and its data easier to manage.

For example, `$data_file` is declared as a global variable that could be used in a program to represent a data file accessible from multiple functions within a program.

Local variables need no prefixing as where they are located within a function, class or block determines the scope of that variable. When you read the program source, you can see the intent for the variable.

Constants are variables that represent values **that never change**. A good practice for representing constants is to name them in **all capital letters**. For example:

```
PI = 3.141592653589
```

or

```
OPERATING_SYSTEM = 'PCLinuxOS'
```

What you name a variable is important. This makes it clear what the variable stands for and where and why it is used.

Keep in mind that variable names cannot ever have spaces, periods, or commas in them as they are used as delimiters separating keywords and variables. Therefore, it is a good idea to use **underscores** as replacements for spaces, periods and commas.

Ruby has two more types of variables, namely **class** and **instance** variables. The **class** variable, as one could guess, is accessible to variables that are part of a **class** as well as variables that have been declared to be of a **class** type. **Instance** variables refer to the variable assigned a particular **class** and are accessed **only from that particular variable**. Think of **instance variables** as the equivalent of **local variables** within classes.

To understand this, we need to know more about classes and instances.

Ruby has class

In object oriented programming, the **class** is a container where data and methods (aka functions and procedures) for working with that data are packaged together.

We have already seen examples of three classes that are part of Ruby. The **Math** class, which we used to calculate square roots, and the **String** class, which allows us to convert strings to other data types, including arrays (which will be discussed later) and integers and floating point numbers. In fact, the String class enables the **string** variable type to exist.

Methods are functions and procedures that are defined within a class. For the **string** class, we have:

- **chomp**: Returns the input string *without* the **newline** character.
- **to_f**: Convert a string variable to a **floating point** number.
- **to_i**: Convert a string variable to an **integer**.

Which is what we learned so far in this series.

The official documentation for current versions of Ruby contains a reference to all classes that come with your Ruby installation. That documentation is located at <https://www.ruby-doc.org>. (Click on **Core API** to get to the **classes**.)

The basic structure of a **class** in Ruby is defined by beginning with:

```
class <name of class>
```

followed by one or more variables that are common to instances of the defined class

```
@@<variable name>=<initial value>
```

and then any methods to be contained in each instance of the class. Methods are defined the same way we defined procedures and functions when programming

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

Ruby using **structured programming** techniques defined in the first two articles of this series, i.e.

```
def <name of method>
```

then place Ruby code for each method here followed by

```
end
```

to indicate the end of each method. Finally, a

```
end
```

is placed here to indicate this is the end of the class definition.

It is a good idea to indicate the end of a method in the defined class by typing:

```
end # name of method
```

and to end the class definition by typing

```
end # name of class
```

By doing so, it makes it clear whether this **end** refers to a method, a class, a loop, a block of code, or even the entire program.

When defining classes, it is customary to provide a method that initializes each instance of the defined class so variables common to instances of the defined class contain a starting value, and all methods contained in each instance function the same way. **If this is not done, the execution of the program will be unpredictable and therefore will be difficult to debug and fix due to the undefined values associated with uninitialized classes.**

The “@@” prefix in the variable names at the beginning of the class definition indicate that the variables defined here are **class variables**, i.e. variables that are common to all instances of the defined class. This means these variables are **global to all instances** of the defined class.

Likewise, a single “@” in the variable name indicates that this is a **instance variable**, i.e. a variable being defined for *that particular instance* and is local **only to that instance**.

Now that we have defined classes...

Once we defined a class, the next thing we need to do is start **using** classes.

Using classes is no different from defining variables. What is different, however, is that **every class** has a method called **new**.

The method **new** is not implemented, but is a part of the Ruby kernel that is responsible for handling objects within a Ruby program.

Arrays are limitless

An array is an example of a class that comes with your Ruby installation. Arrays are a form of a data container that stores a collection of any type of object, including other classes and arrays. Yes, this is a recursive definition, but then the idea of placing arrays inside of arrays is recursive as well.

In traditional programming languages, arrays are usually collections of **one type of data**. Ruby, however, allows arrays that are collections of **any data type**, and **data types can be mixed**, since *everything in Ruby is an object*. **As a result, arrays are objects as well!** (This is what makes Ruby arrays **limitless**.)

Think of the possibilities of how arrays can be used.

Ruby Arrays Greatest Hits

Ruby arrays can be used to simulate a database record by defining the custom record type as an array. For example, the following two statements do the same thing:

```
customer_record = Array.new  
customer_record = []
```

This is possible since Ruby arrays can contain a mixture of data types.

For this example, let us define a customer record with the following fields:

- Last Name (40 characters)
- First Name (40 characters)
- Address (two lines of 64 characters each)
- Zip Code (10 characters, Alphanumeric)
- Phone Number(s) (12 characters)
- e-Mail address (80 characters)

In Standard (UCSD) Pascal, this data record would be implemented as follows:

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

type

(* Record type for customer information here in UCSD Pascal *)

```
customer_record : record
  lastname : array [1..40] of char;
  firstname : array [1..40] of char;
  address1 : array [1..64] of char;
  address2 : array [1..64] of char;
  zipcode : array [1..10] of char;
  phone_number : array [1..12] of char;
  email : array [1..80] of char;
end;
```

For Turbo Pascal versions 4.0 and later, the same record would be implemented as follows:

uses String;

{ Record type defined here for Turbo Pascal 4.0 and later }

```
customer_record : record
  lastname : string[40];
  firstname : string[40];
  address1 : string[64];
  address2 : string[64];
  zipcode : string[10];
  phone_number : string[12];
  email : string[80];
end;
```

Turbo Pascal 4.0 and later, as well as FreePascal contain modules called **units**. Ruby implements modules the same way as we will see later in this series. Without the **uses** statement, this record could be implemented as:

{ Record type defined here for Turbo Pascal 4.0 and later }
{ alternate implementation }

```
customer_record : record
  lastname : packed array [1..40] of char;
  firstname : packed array [1..40] of char;
  address1 : packed array [1..64] of char;
  address2 : packed array [1..64] of char;
  zipcode : packed array [1..10] of char;
  phone_number : packed array [1..12] of char;
  email : packed array [1..80] of char;
end;
```

As we can see, Ruby has an advantage of the fact that there is **only one**

implementation of Ruby as opposed to several implementations of Pascal, despite UCSD Pascal being the **official standard Pascal**.

Did you know that Pascal was originally developed as a **write once run anywhere** system (not unlike Java)? UCSD Pascal came with a editor, compiler, and a p-code virtual machine, which interpreted compiled bytecode. This meant that machines that UCSD Pascal programs compiled on one machine would be able to run on **any** machine that had a **p-code** virtual machine installed.

Version 2 of UCSD Pascal was available for the Apple II series machines (as Apple Pascal), the Commodore PET series as well as Commodore 64 and 128 machines, Texas Instruments TI99/4A with the p-code cartridge plugged in), NEC's Advanced Personal Computer (on 8-inch floppies or those who remember using disks that big), and even the early IBM PC, XT and AT models.

On September 10, 2018, Kenneth Bowles, the creator of UCSD Pascal, passed away at age 89. There is a memorium at UCSD's website on this.

To finish the Ruby implementation of this record data type, here is one suggestion:

=

The customer data type being defined as an array of seven fields, with indices numbered 0 through 6.

In Ruby, array indices always start at zero instead of one. This is true for arrays in C, C++, and Java.

The indices are as follows:

0 : Last name of the customer
1 : First name of the customer
2 : Street address
3 : City and state
4 : Zip code
5 : Phone number
6 : e-mail address1

=

```
customer_type = %w(last_name first_name street_address city_state zip phone e-mail)
```

where each of the parameters would contain **actual data** instead of the placeholders shown in the statement.

Ordinarily, when entering actual strings inside arrays, Ruby expects double

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

quotes for each parameter. The `%w()` created an array of words **without quotes and with a space separating each word** in the array. Each word in the `()` is an entry in the array.

*For the `%w()` to work as expected, each entry must contain **exactly one word** with one space between each word.*

Otherwise, all string entries in the `[]` for defining arrays **must be in double quotes**.

Since we just defined `customer_type` as an array (that simulates a record type), there is no reason we cannot define an array of `customer_type` objects. So, we define a new object called `customer_list` and add a `customer_type` to that list.

```
customer_list = Array.new
customer_list << customer_type
```

The `<<` makes it obvious that a new object of `customer_type` is being added to the array we defined as `customer_list`. Ruby provides a `push` method that does the same thing, so we could have written the last two lines of code as:

```
customer_list = Array.new
customer_list.push customer_type
```

and got the same result. I prefer the former (the one with the `<<`) as a good practice.

Methods for accessing arrays

The `size` method returns an **integer** that tells us how many elements are in a given array. For instance:

```
number_of_customers = customer_list.size
```

tells us how many customers are in this list. The following statements are equivalent:

```
selected_customer = customer_list[0]
selected_customer = customer_list.first
```

Both of these statements return the first customer in the list. Remember that the indices of an array **always start at zero, not one**. (This is true in C, C++ and Java as well.)

The next statements are equivalent as well:

```
selected_customer = customer_list[(customer_list.size-1)]
selected_customer = customer_list.last
```

The `size` method returns the **number of elements that are in the array**. However, the index associated with the last element in the list *is always one less than the size of the array* due to the indices of an array **always starting at zero**. Hence, the last element will have an index of `customer_list.size-1`.

Arrays as stacks

One common data structure used in programming is a stack. A stack is a list where elements are added to the stack by pushing them or removed from the stack by popping them.

We already know how to push elements onto a stack with the `push` method as follows with these two equivalent statements:

```
customer_list << customer_type
customer_list.push customer_type
```

We can remove elements from the stack with the `pop` method. Unfortunately, the only way to do this is with this statement:

```
deleted_customer = customer_list.pop
```

(The `>>` has not been implemented as a `pop` method as of the current version of Ruby.)

Ruby arrays are double ended

There are two more methods for data manipulation in Arrays. The `shift` and `unshift` arrays add and remove elements in the array **from the front end, i.e. the element with the index of zero** rather than the end as is the case with a normal stack.

```
customer_list.unshift customer_type
```

adds `customer_type` to the front of the list making it the first element in the array `customer_list`. To get rid of that `customer_type`, we include the following code:

```
deleted_customer = customer_list.shift
```

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

The **shift** method assigns the first element in **customer_list** to **deleted_customer**, then deletes that customer from the list.

Searching made simple

The **include?(element)** method (the question mark is part of the method name) searches the array for the existence of **element** and will return **true** if that element is found, and **false** otherwise.

Printing the Array contents

The **each** method allows operations on elements in the entire array *without the use of a for statement*.

For example, to display the contents of the entire array, we would use the following statement:

```
customer_type.each {|item| puts item }
```

This statement requires **{}** as this is technically a block of code. The **each** method here could have been written as:

```
for item in 0..(customer_type.size-1)
  puts customer_type[item]
end
```

Given the examples we used throughout this article, we could print the entire customer list as

```
for index in 0..(customer_list.size-1)
  customer_list[index].each {|item| puts item }
```

Records have Class

Let us revisit the record definition.

```
=
The customer data type being defined as an array of seven
fields, with indices numbered 0 through 6.
```

In Ruby, array indices always start at zero instead of one. This is true for arrays in C, C++, and Java.

The indices are as follows:

```
0 : Last name of the customer
1 : First name of the customer
2 : Street address
3 : City and state
4 : Zip code
5 : Phone number
6 : e-mail address1
```

```
=
```

```
customer_type = %w(last_name first_name street_address city_state
zip phone e-mail)
```

We could redefine this record type as a **class**, then create an array of objects with that class.

```
class Customer
```

```
  @@last_name = ""
  @@first_name = ""
  @@street_address = ""
  @@city_state = ""
  @@zip = ""
  @@phone = ""
  @@e_mail = ""
```

```
  def initialize(last, first, street, city_and_state, zipcode,
phone_number, email)
    @last_name = last
    @first_name = first
    @street_address = street
    @city_state = city_and_state
    @zip = zipcode
    @phone = phone_number
    @e_mail = email
  end # initialize
```

```
end # customer_type
```

The **initialize** method should be defined for **every class you create** ensuring that there is valid data contained within the class as each instance of the class is defined. **The initialize method is automatically executed** every time an instance of the class is created.

What we have here is the foundation for what we can do with objects of type **Customer**. Simply add methods to the class by defining them inside the **class** definition and placing those definitions after the **initialize** method.

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

Consider the **each** method when this record was defined as an array.

```
customer_type.each {|item| puts item }
```

When this statement is executed, each field in the record is printed on a separate line.

```
customer's last name
customer's first name
Street address
City and state
Zip code
Phone number
e-mail address
```

Let us now compare side by side each field in the class declaration to the fields in the array declaration.

Class declaration	Array Declaration
@@last_name	customer_type[0]
@@first_name	customer_type[1]
@@street_address	customer_type[2]
@@city_state	customer_type[3]
@@zip	customer_type[4]
@@phone	customer_type[5]
@@e_mail	customer_type[6]

It is apparent that implementing record types as a **class** is better than implementing record types as an **array**.

This example could be used to create a simple address book. By implementing this data record as a **class**, this allows for expansion by simply adding fields to the variable declarations. In addition, **subclasses** could be created that use the data within this class rather than replicating the data fields in its own class. This is an example of the concept of **inheritance** in object oriented programming. This could be done with the record implemented as an array, but the process needed to implement this inheritance would be **quite complicated and prone to errors**, as one would have to track the smaller details in the implementation, not to mention this would make the resulting code difficult to read and understand.

String Manipulation

When we defined the record type in the previous examples, the fields of each record were of the **string** type. Strings are a type of array, namely an array of

characters. Like arrays, strings can be of any size. However, strings are one dimensional by themselves.

Ruby has a concept called **string interpolation**. Remember the first examples we used to output data to the screen?

With the **puts** and **print** methods, we can output quoted strings directly. **String interpolation** is coded *within an output string* by placing a hashtag (#) followed by braces as follows: **#{}.**

What goes between the braces, however, could be the variable name itself, or variable slightly modified with Ruby code embedded between the braces. This code is executed using the **to_s** method, or the built in function within the string class that converts any object to a string object, including arrays.

Let us look at an example of a string:

```
dissertation = "Creationist Kent Hovind Doctoral Dissertation"
```

I first heard about this document from a parody (hosted at Amiright.com) of the Disney classic song "Supercalifragilisticexpialidocious" called "Creationist Kent Hovind's Doctoral Dissertation".

At that time, I did a web search on this infamous document, and then subsequently downloaded the document (PDF) for later reading. It is good for entertainment, **but definitely not academic by any means.**

The fact that this document was actually hosted on Wikileaks should be sufficient enough to create doubt about its integrity. If you think this document is unbelievable, check out the article on Wikipedia about Kent Hovind himself.

To print this object out, we could use:

```
puts dissertation
```

or this:

```
puts "#{dissertation}"
```

and get the same result. We can also display *part of a string* by specifying a range with the object name, not unlike how we access elements in arrays.

```
puts "#{dissertation[0..7]}"
```

would display

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

Creation

As with arrays, strings have indices that start at zero (0), representing the first character in a string. What is different about strings is that negative index values are possible with strings, *but not with arrays*.

An index of **-1** refers to the last character in a string (which could be a newline character, or **“\n”**), whereas an index of **-2** refers to the next to last character in that same string. The latter would be useful for assigning a truncated string (not unlike the **chomp** method normally used with **gets**) to a variable.

If we had written

```
puts "#{dissertation[0..7]}"
```

As

```
puts "#{dissertation[0,7]}"
```

the result would display of the word **Creatio**, or the word “Creation” *without* the trailing **“n”**. To get the same result, we would need to write the latter as:

```
puts "#{dissertation[0,8]}"
```

In this form, the string displayed starts at the index of **zero**, or the beginning of the string. The second parameter in this form tells Ruby **how many characters to write** to the screen, which here would be eight. If we were to write that statement as:

```
puts "#{dissertation[24,8]}"
```

we would get

Doctoral

for a result on the screen. We already know how to display partial strings. But we can also delete or replace text within the range by simply assigning a string to **that range** within the string rather than the entire string. (Assigning a null string to a range *deletes* all characters within that range.)

Since we can replace or delete text from a string, it also makes sense that we should be able to search within a string. The **include?** method we used in Arrays applies to strings as well.

```
dissertation.include?("Doctor")
```

would return **true** as **Doctoral** contains the word **Doctor**.

Searches here are **case sensitive**. This means that **dissertation.include?("doctor")** would return **false** because **“doctor”** is not the same as **“Doctor”** when it comes to this method. To correct this, we should use either the **uppercase** or **downcase** methods such as **dissertation.downcase.include?("doctor")** to ensure the accuracy and reliability of the method.

All we know at this point is that the word “Doctor” exists in the string. But we do not know the exact location where “Doctor” exists in the string. For that, we use the following:

```
dissertation.index("Doctor")
```

which will return the index where *Doctor* exists in the string.

Filling in Strings in Ruby

There are times when we want a string to be a specific length, for example, we want to display a integer as a binary number. 16 bit signed integers have a range from -32768 to +32767. To display these numbers as binary digits, we need to define a string of 16 characters in length.

Suppose we want to display the binary digit **“10110”** as a 16-bit number. Since there are five binary digits in this number and we need 16 binary digits, we need to pad the binary number with **11** zeroes.

In Ruby, this accomplished as follows:

```
binary_number = "10110"  
binary_number.rjust(11,"0")
```

The **rjust(count,filler)** method prepends the string with **count** instances of the character represented by the **filler** parameter. For this example, **10110** becomes **000000000010110**, which is a 16 digit representation of the binary number **10110**.

Ruby provides a **ljust(count,filler)** that appends the string with **count** instances of the character represented by the **filler** parameter. If we used **ljust** instead of **rjust**, **10110** would have become **1011000000000000**, which is clearly not the same number as **000000000010110**.

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

There are other cases where this would be appropriate. Typically, when computers print out checks (before the days of direct deposit), the amount of the check would be prepended by a series of asterisks as a means of preventing fraudulent changes in the amount of the check.

String Comparison without Case Sensitivity

When strings are compared, they are compared character by character, and that inherently means the strings have to exactly match, including upper and lower case. This means that “**A**” is not the same as “**a**” even though we can see they are the same letter.

Ruby has two methods that can be applied to strings that solve this problem. Either compare the string in **all upper case letters** or compare the string in **all lower case letters**.

The method **upcase** converts the string to all upper case letters (where applicable). Likewise, the method **downcase** converts the string to all lower case letters.

Take it All Off

There are times when we need to use only the actual text and not the extra characters such as tabs, spaces, carriage returns and newline characters. The **strip** method does just that with any string containing whitespace characters before the beginning of the string and after the end of the string. This is useful when using data files that often contain extra whitespace to make the file readable to people.

The **lstrip** removes any whitespace from the beginning of the string to a non-whitespace character in the string. The **rstrip** method removes any whitespace from the end of the string to the first non-whitespace character in the string.

For example:

```
header_line = "   This line is just seven words long   "
header1 = header_line.lstrip
header2 = header_line.rstrip
header3 = header_line.strip
```

The variables **header1** and **header3** appear to be the same text when printed, but the extra spaces are printed from **header1** whereas there are no extra spaces after **header3**. So **header1** will print

```
|This line is just seven words long      |← Cursor is here
```

For **header2**, the following will print:

```
| This line is just seven words long|← Cursor is here
```

and finally, for **header3**, the following will print:

```
|This line is just seven words long|← Cursor is here
```

The **|** characters separate where the cursor on the screen starts and there the text display ends. They are not part of the output and are shown here to show what gets output.

Give that String Arrays

The **split** and **join** methods convert between arrays of characters (or strings) and the strings they make.

When **split** is used without a parameter (usually a character that separates values), **split** defaults to using a space as a separator.

```
header_line = "This line is just seven words long"
```

Let us split this line into an array of seven words.

```
header_array = header_line.split
```

Now, **header_line** is an array of seven words.

```
header_array = ["This","line","is","just","seven","words","long"]
```

This line can be put back together again, using the **join** method.

```
header_line = header_array.join
```

Again, the space is used by default to separate values in the array. Using the following splices the values into one continuous string.

```
header_line = header_array.join("")
```

The parameter supplied here is a Null character, or nothing added between values when the string is created.

Ruby Programming Language: Data Handling with Variables, Classes, Arrays and Strings

Next Time: We Build A Useful Module

We have enough now to be able to create a set of classes we can package as a **library** (or **module**) in Ruby terms. There is more to be done with strings and arrays, but I want to start a useful project with Ruby: **Building our own text editor**. (If we are going to learn a programming language, why not do something useful with it?)

In my senior year in high school (Fall 1984 to Spring 1985), I took a course in Business Machines. Each computer in the classroom was an Apple II+ connected to a Epson FX-80 printer. There was also one Apple IIe machine similarly equipped.

In the same side of the building was another classroom filled with IBM Selectric typewriters, where I took a course in typing (I could type 32 words per minute) that fall semester.

The main computer lab, located on the lower level floor of the school's library consisted of Apple II+ and Apple IIe machines, with two of the latter machines had Epson FX-80 printers connected.

For the word processing program, the school had **Broderbund's Bank Street Writer** which displayed text as shape tables for the font (to accommodate 64 characters per line as the text mode of the Apple II+ only allowed 40 characters per line).

Shape tables are binary coded vector graphics used with AppleSoft's DRAW command.

At the time, I had to use that product to do the assignments for the Business Machines class. The product was available to all machines in the high school at that time. Since I did not particularly like that product, (After all, how many businesses *actually used Bank Street Writer in their offices?*) I decided to **build my own word processing software** to use for other assignments.

This was easily accomplished in AppleSoft BASIC, and I wrote it for the Apple IIe to use the 80-column feature for text display.



PCLOS-Talk
Instant Messaging Server

Sign up TODAY! <http://pclostalk.pclosusers.com>



The PCLinuxOS Magazine Special Editions!



The PCLinuxOS Magazine Special Editions include:

- Windows Migration Guide (September 2013)
- Enlightenment Special Edition (May 2011)
- The KDE 4 SC Special Edition (Fall 2010)
- Gtk Lightweight Desktops: Xfce & LXDE Special Edition (November 2010)
- Openbox Special Edition (March 2012)
- Command Line Interface Intro (October 2010)

Get Your Free Copies Today!

Forgotten Wifi Passwords? Here's How To Find Them

by Paul Arnote (parnote)

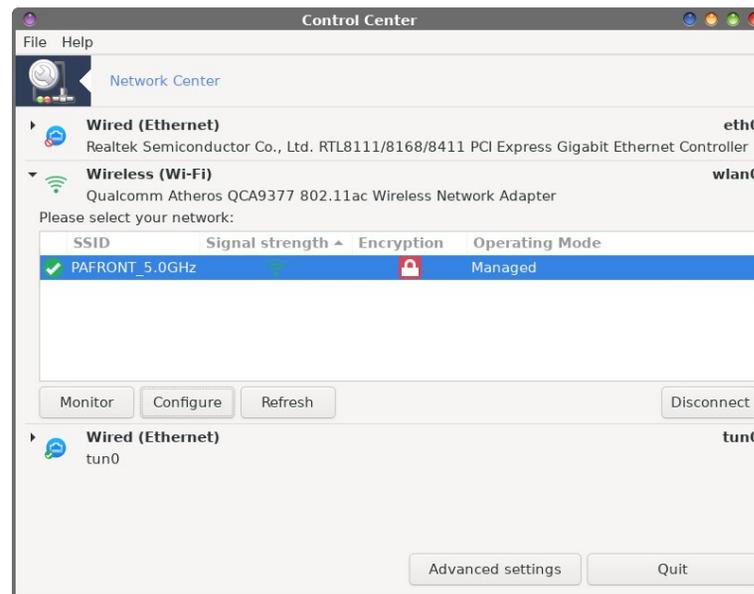
PCLinuxOS, and Linux in general, is pretty good about remembering the passwords for our wifi connections. In fact, it may be too good. Usually, once we enter a valid password for a particular wifi connection, we rarely – if ever – have to enter it again. That ability to remember wifi connection passwords can also aid us in forgetting them, which can lead to some frustrating moments when it comes time to recall them.

Maybe you just forgot the password(s). And, who could blame you if you did? Or, maybe you want or need to share your wifi connection with guests, but can't remember the password. Hopefully, you're using complex passwords that can't easily be guessed, either. Plus, let's hope you're using WPA/PSK, and not WEP, for wifi security. WEP can typically be "broken" in less than 60 seconds by a determined and modestly intelligent hacker. While WPA/PSK is not immune from being hacked, doing so requires a LOT more work. A typical hacker will most likely move on to an easier target.

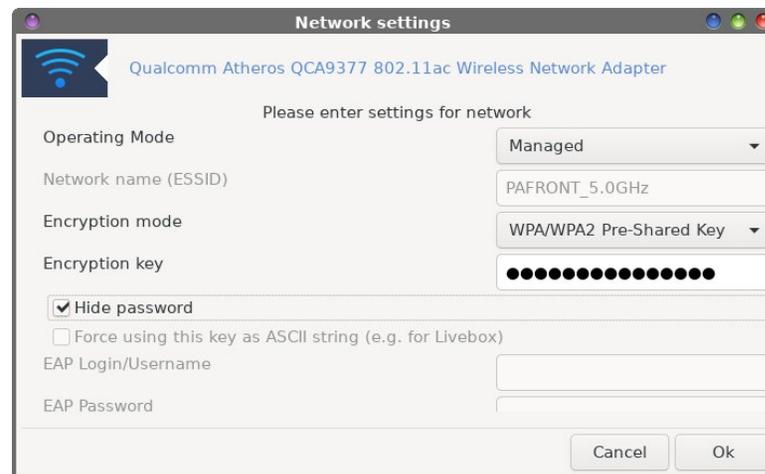
Fortunately, there are a couple of ways to "rediscover" your saved wifi connection passwords. Let's start by taking a look at the graphical way to recover them.



Open the PCLinuxOS Control Center. Click on the "Network Center" icon.

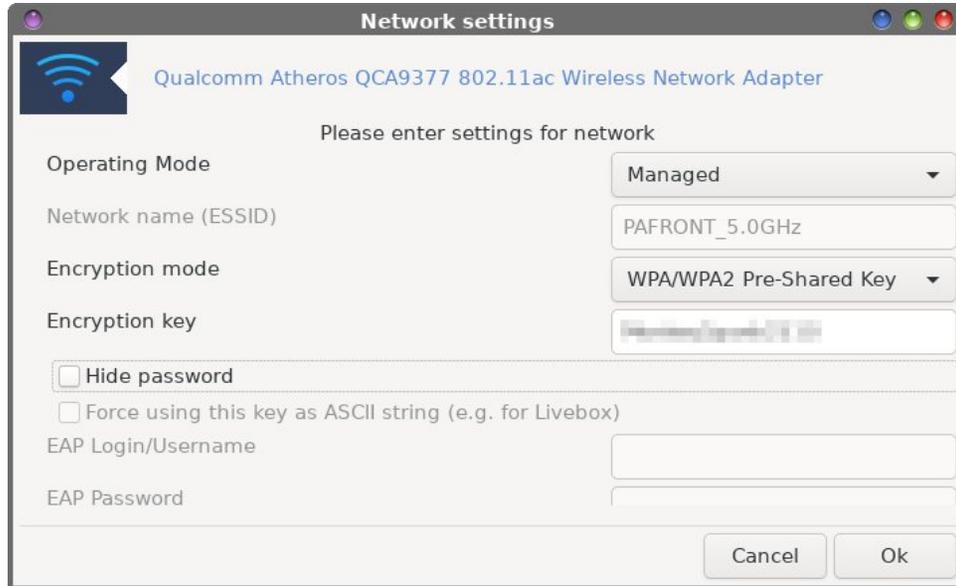


Now, click on the caret next to the connection you want to use. In this case, it should be your wifi connection. Select the wifi connection, and then click on the "Configure" button.



Forgotten Wifi Passwords? Here's How To Find Them

The fourth item down in the new window is labeled "Encryption key." Just beneath that, on the left side of the window is a checkbox labeled "Hide password."



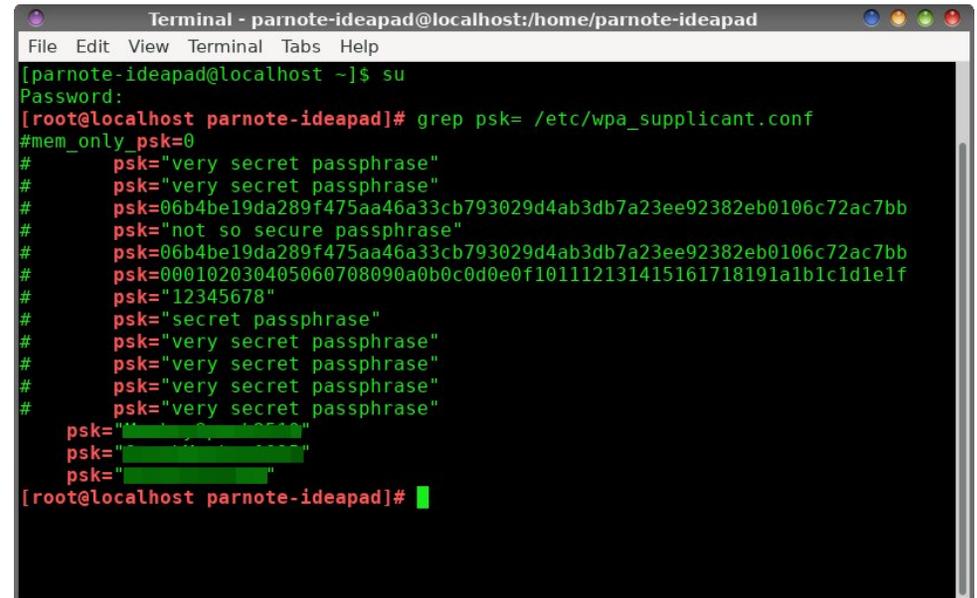
Uncheck the checkbox, and the password for your selected wifi connection will be displayed. (My information is blocked from your view, for obvious reasons).

Of course, this being Linux, there is more than one way to "rediscover" your wifi connection passwords. Let's take a look at how to do it from the command line.

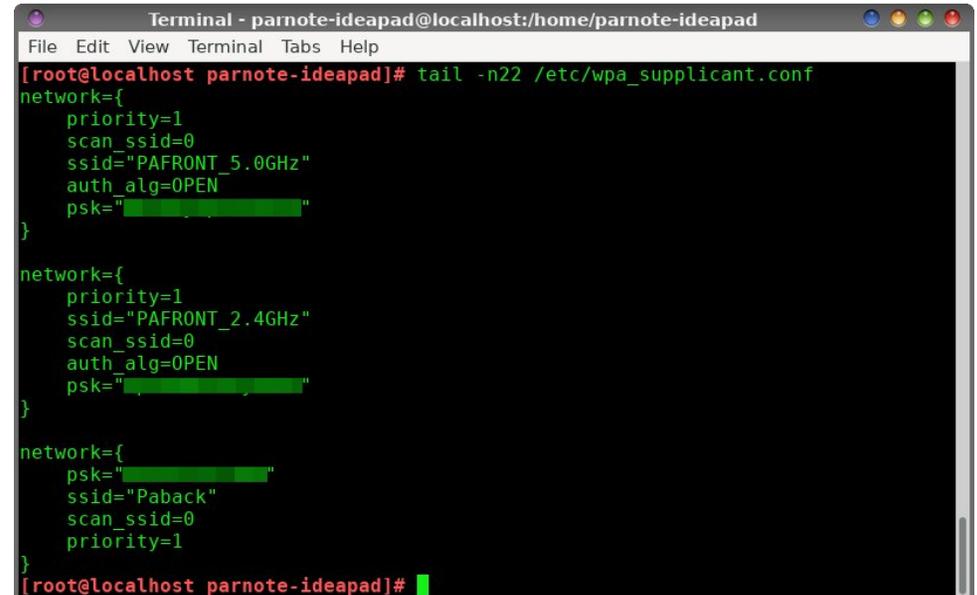
```
[parnote-ideapad@localhost ~]$ grep psk= /etc/wpa_supplicant.conf
grep: /etc/wpa_supplicant.conf: Permission denied
```

Of course, you can't access the information as a regular user, so you will have to issue the command as the root user.

As the root user, enter `grep psk= /etc/wpa_supplicant.conf` at the command line.



Displayed (and blocked out for your viewing pleasure) will be all of the wifi connection passwords you've used at the bottom of the output. The passwords are not listed with the particular network they are associated with, at least by using this command.



Using the command `tail -n22 /etc/wpa_supplicant.conf` (as the root user) will give me not only the wifi connection passwords stored on my computer, but also give me the SSID of the wifi network connection that each password is associated with. You may need to use a number larger than 22 (number of lines at the end of the file to display) if you have more wifi connections stored in `wpa_supplicant.conf` than the three that I have stored.

Now if you're thinking that there is an inherent security liability from storing the wifi connection passwords in unencrypted text files, you might be right if access was open to all who wished to view it. But, the information is only available to the root user. So, if anything, this illustrates that you need to be careful about who has access to the root account. Graphically or from the command line, the information is only accessible by the root user.

As far as I know, none of this information is useful to help you crack or hack a wifi connection password you do not already have access to or been granted access to. For that, there are other tools you can use or try, with variable success.



PCLinuxOS-Cloud secure private simple-to-use
Sign up TODAY! pclusers.com/services-signup.php

Screenshot Showcase



The screenshot shows a Linux desktop environment with a dark theme. The wallpaper features a silhouette of a tree on a small island with a large, bright moon in the background. On the left side, there are icons for 'Trash', 'Documents', and 'Current DATA'. On the right side, there is a system information panel titled 'Linux 5.0.7-pclos1' with the following data:

Processor	Usage
0.50GHz core1	1%
0.50GHz core2	1%
2.25GHz core3	2%
2.25GHz core4	2%

Below the processor information, it says 'conky-2'. Further down, it lists system resources:

RAM:	719MiB/7.67GiB
HDD:	12.1GiB/49.4GiB
DISK I/O:	0B
NETWORK:	0B
SYSTEM TEMP:	50 C

At the bottom, there is a taskbar with a 'Menu' button, several application icons, and system tray icons including volume, network, and a clock showing 11:02. A battery icon shows [N/A].

Posted by Yankee, May 9, 2019, running LXDE.

DOS GAMES ARCHIVE
WWW.DOSGAMESARCHIVE.COM

Donate NOW



Short Topix: Google, Other Tech Giants Buying Up Internet Undersea Cables

by Paul Arnote [parnote]

IE: Yet Another Security Flaw



Yet another [security flaw](#) has been found in the “internet-browser-that-wouldn’t-die-even-though-Microsoft-wishes-it-would,” Internet Explorer. The last version of Internet Explorer was released in 2013, six years ago. Despite being replaced by Microsoft’s newer browser, Edge, Microsoft still maintains multiple versions of Internet Explorer to support legacy applications. In fact, in the ensuing six years, even Edge has seen some significant changes, having been rebased just this year on Google’s open source Chromium browser.

And this security flaw is a doozy. Users could be at risk just by having it installed. The security flaw comes from MHT files, which are IE’s web archive

format. Windows uses IE to open MHT files, by default. But you don’t even have to open IE to be affected. If you download a MHT file sent through chat or email, hackers can spy on Windows users and even steal their local data. That data can include contacts, online transactions, and other private and personal data that users are typically lax about security on.

While Microsoft has said that it “will consider” a patch for the vulnerability in a future update, it isn’t assured that one will be coming. Therefore, millions of Windows users are at risk until a patch is released, if it’s even coming at all. So, the advice to Windows users is to either [turn off](#) Internet Explorer, or point to another program that can open MHT files. Of course, it might be wise to avoid downloading any MHT files, as well.

I don’t know about you, but it’s things like this that make me very glad to be a PCLinuxOS user!

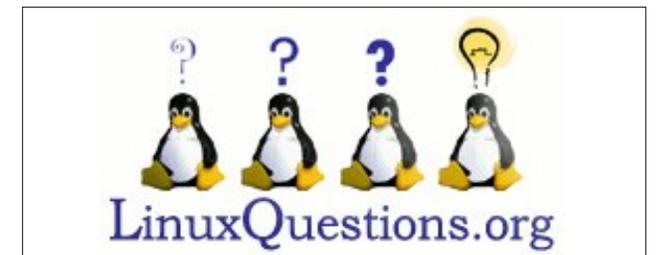
Make Firefox More Efficient By Disabling Lazy Loading



By default, Mozilla’s Firefox web browser has “lazy loading” turned on. Not sure what “lazy loading” is? Let’s explain.

Let’s say you have a number of “pinned” tabs that are loaded when Firefox starts up. Or, let’s say that you have Firefox setup to remember your open tabs when it closes, so they can all be re-displayed when Firefox is relaunched. With “lazy loading” turned on, the content of each tab is not loaded until you click on each particular tab. This initially improves Firefox’s loading time, and some users are just fine with this behavior. But, some users want to work more efficiently by having the content of all tabs already preloaded when it is needed, instead of waiting for each tab to load its content.

Fortunately, it’s relatively easy to turn off “lazy loading.” Open a new tab in Firefox, then type **about:config** into the address bar. Agree to “be careful” and accept the risk (if you’re a long time Firefox user, you probably already know the routine), and then type **browser.sessionstore.restore_on_demand** on the search bar in the window that subsequently appears. The default value will be set to “true.” Double click on the “true” setting to change it to “false.” Close and then restart Firefox. Now, all of your tabs will load just as soon as Firefox loads, and you’ll no longer have to wait for each of them to load when you click on each individual tab.



Short Topix: Google, Other Tech Giants Buying Up Internet Undersea Cables

Google, Other Tech Giants Buying Up Internet Undersea Cables

I recently read an alarming [article](#) on VentureBeat that I had never really thought about before. There are over 700,000 miles (1,147,550 km) of submarine (undersea) cables in use today. Historically, those cables have been owned by groups of private companies, comprised mostly of telecom companies. But, 2016 saw a new group jump into the submarine cable arena: content providers.

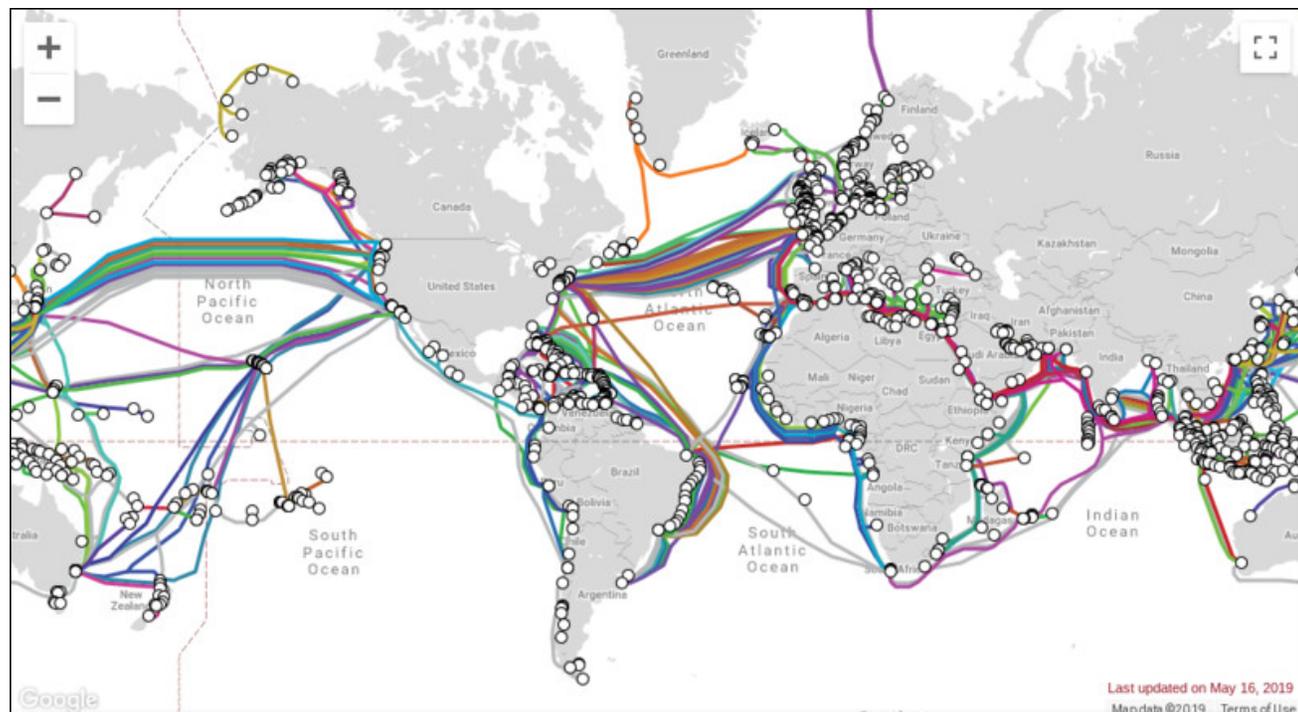
Google leads the way, with the direct/whole ownership or part ownership of nearly 65,000 miles (106,560 km) of submarine fiber optic cable. That's enough submarine fiber optic cable to wrap around the Earth's equator 2.5 times, with a few thousand miles of cable to spare. This past February, Google announced plans to move forward with the Curie cable, a new 6,400 mile (10,476 km) submarine cable from Los Angeles, Calif. to Valparaiso, Chile.

Google isn't alone in its pursuit of ownership of submarine fiber optic cables, either. Other tech giants are also clamoring for their share of the pie. These companies include Amazon, Microsoft, Facebook, and others. You can view an interactive map from TeleGeography of all the major submarine fiber optic cables [here](#). The interactive map will display the owner of each cable. Just click on the cable name in the sidebar on the right side of the map.

As mentioned earlier, historically these submarine cables have been owned by private telecom companies. Back when they were being installed, we didn't worry about privacy concerns. We trusted those companies, rightfully or wrongly. But now, these tech giants are buying up the literal backbone of the internet. Given their utter disregard for users' privacy and their cavalier attitude towards users' private information, well ... therein lies the concern that leads to the sense of alarm.

As the owners or partial owners of these information conduits, carrying massive amounts of data, what's to prevent them from siphoning off even more of our private information that they already don't possess or respect? These tech giants have done exceptionally little to earn our trust over the years, and have behaved especially poor in recent years. Then, there's the rumors and speculation of just how much the tech giants are "in bed" with the three and four letter government spy agencies.

Without a doubt, the submarine fiber optic cables are necessary not only for growth of the internet, but also for shaping the internet of tomorrow and beyond. I'm just not sure that the giant tech companies are the best caretakers/custodians of these information conduits. While I realize that this may sound like rank paranoia, or that it may make me seem like a conspiracy theorist, given their track record, they haven't given any of us enough reason to trust them with our privacy and private data. In fact, given their recent "performances" in this area, they have provided ample cause for doubt and suspicion. So not only are they sucking up our private data faster than a Hoover vacuum cleaner on steroids, they are now going to OWN the means of getting that data from point A to point B. Users would be negligent to not be concerned.

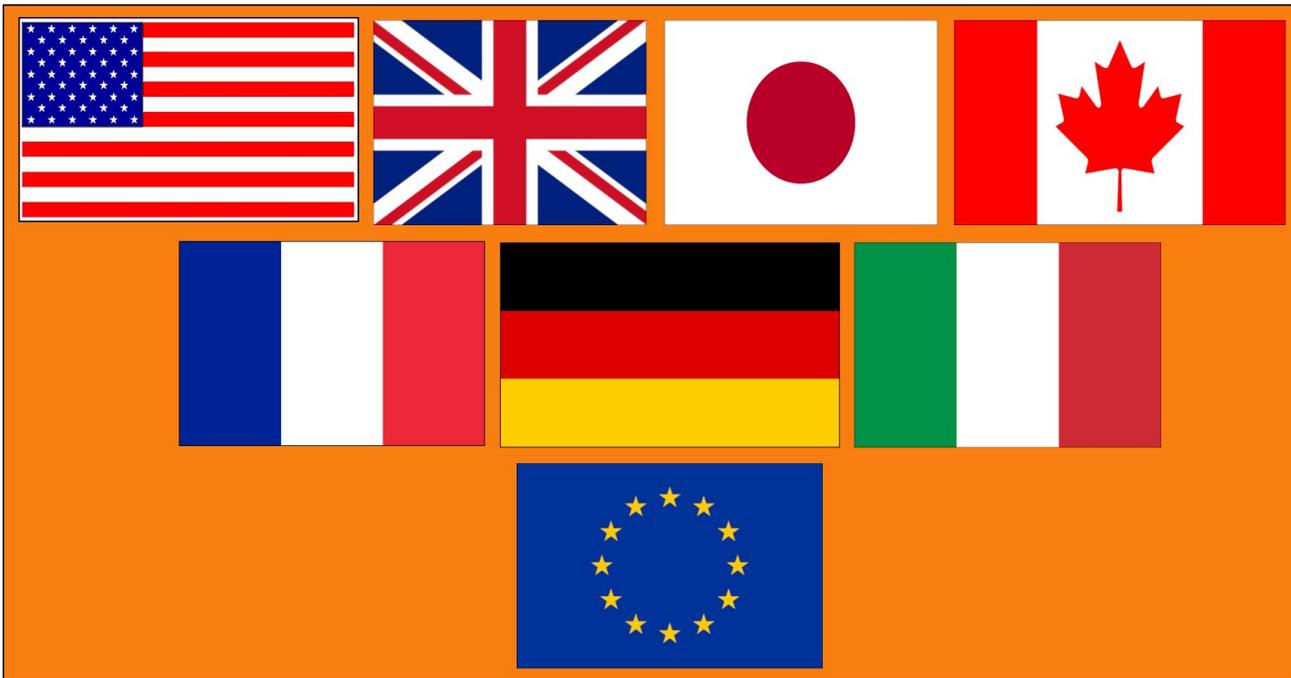


G7 Countries Negotiate Tech Regulation Charter

You've heard of the G7 ([Group of 7](#)) countries, right? The group is made up of the United States, United Kingdom, Japan, Canada, France, Germany and Italy. The EU also has a representative. Started back in 1975, these countries have the seven largest advanced economies in the world. Together, they represent 58% of the global net wealth, and more than 46% of the global gross domestic product (GDP). The G7 meets at least annually to discuss economic policies.

So, according to an [article](#) on TechCrunch, the digital ministers/leaders of the G7 countries met on May

Short Topix: Google, Other Tech Giants Buying Up Internet Undersea Cables



15, 2019 to discuss an upcoming charter on toxic content and tech regulation at large. The plan is for the G7 member nations to sign the charter at the upcoming G7 Summit in August, to be held in Biarritz, France. Additionally, officials from Australia, India and New Zealand participated in the meeting.

From the article:

“Everyone has to deal with hateful content,” France Digital Minister Cédric O said in a meeting with a few journalists. “This industry needs to reach maturity and, in order to do that, we need to rethink the accountability of those companies and the role of governments.”

One of the problems to overcome is how some countries want protections for free speech, while others are pushing for more regulation. “Hateful speech” won’t be precisely defined, but a set of guidelines/principles to follow should help countries make laws to address this issue.

We’ll have to wait until August to see how all of this plays out.

Ransomware Holds City Of Baltimore Hostage



The city of Baltimore, Maryland had to take several of its systems offline on May 7, 2019, after their

computer systems were victimized by a nasty ransomware virus, according to an [article](#) on ArsTechnica. More than two weeks later (at the time this article was written in late May), Baltimore’s systems remain offline.

The ransomware attack by the aggressive RobbinHood ransomware is similar to another attack in April that hit Greenville, North Carolina. The FBI, who is helping investigate the ransomware attack, have identified the ransomware as a new variant of the RobbinHood ransomware that has emerged within the past month.

In the past, the RobbinHood ransomware has not been able to be spread across a network. Rather, the software typically has to be installed on each infected computer by a user with administrative access to that computer. It is spread to individual machines via a psexec system internal call, and/or by compromising the domain controller. Additionally, RobbinHood requires that a public RSA key already be present on the target computers in order to begin encryption of the computer’s files. Thus, the attacker would have had to deploy the ransomware in a series of multiple steps.

Here’s an excerpt from the ArsTechnica article that illustrates just how nasty this ransomware is:

*Before it begins encryption, RobbinHood malware shuts down all connections to shared network directories with a **net use * /DELETE /Y** command and then runs through 181 Windows service shutdown commands—including the disabling of multiple malware-protection tools, backup agents, and email, database, and Internet Information Server (IIS) administrative services. That string of commands—which starts with an attempt to shut down Kaspersky’s AVP agent—would create a lot of noise on any management system monitoring Windows systems’ event logs.*

This isn’t the first go-around for Baltimore in dealing with a ransomware attack. In March, 2018, the city’s 911 system was attacked by ransomware when the

city's IT department was doing some maintenance work on the city's firewall. That work briefly left a few holes in the city's firewall, which attackers found and exploited, probably by an automated scan. The firewall holes (and maintenance) were only present for about four hours before the ransomware attacked the system.

It's stories like this that make you scratch your head and wonder what IT departments and CIOs are thinking by putting the operating system with the WORST security ever in charge of mission critical systems. It's stories like this that make me glad to be a Linux user.



Like Us On Facebook!
The PCLinuxOS Magazine
PCLinuxOS Fan Club



Screenshot Showcase



Posted by Mr Cranky Pants - YouCanToo, May 3, 2019, running KDE.

Donate To PCLinuxOS

Community Supported.
No Billionaires/Millionaires.
No Corporate Backing Or Funding.

Click [here](#) to make a one-time donation through Google Checkout.

Or, click one of the amounts down below to make a monthly, recurring donation.



PCLinuxOS Family Member Spotlight: BillDjr

As told to YouCanToo



What is your name/username?

Bill Dawson Jr. (BillDjr)

How old are you?

Feels like 1,000. I'm 56.

Are you married, single?

I've been very happily married for 20 years. My wife is a saint.

How about Kids, Grandkids (names and ages)?

I have 2 kids. Liam, age 15 and Chorin, age 10. Best kids ever.

Do you have pets, what is your favorite?

We have a dog named Buddy. He has a supernatural gift to piss me off.

Are you retired, still working and if working, what do you do?

I'm on a disability income. Always on the hunt for things to do.

Where do you call home? What is it like? IE: weather, scenery

Ottawa, Ontario, Canada. Yes, winters are brutal. We all live in igloos. Summers are equally brutal. Way too hot and way too humid. We are blessed with plenty of greenery everywhere we go.



Parliament Hill, Ottawa



Embassy Tower of Taiwan, Ottawa

Where did you go to school and what is your education level?

I went to Merivale High School in Ottawa, but missed graduating by 3 credits. I got my real education at the school of hard knocks.

What kind of things you like doing? hobbies, travel, fishing, camping?

It frustrates the hell outta me sometimes and hurts my brain, but I really enjoy programming. Can't wait



The Maman Statue

until I have enough experience with python to start using that full time instead of the IDE for BASIC I've been using.

I am totally fascinated by quantum mechanics. The implications on the nature of reality are earth-shattering. I'm also one of those "red pill" people, if you know what that is ... always going down many rabbit holes at the same time. That, too, can be earth-shattering.

I've only ever traveled once, to Cozumel with my wife. We absolutely adored the place. Sadly, finances have always been an issue for us, so more traveling doesn't seem to be in the cards.

I'm an ex-wanna-be rock star, and recently my son Liam has begun putting together techno and dubstep music tracks on his laptop. That boy's got some real talent. We have started having lengthy discussions about the nuances of music and his tracks in particular, which gets us both really fired up. He woke up the musician in me, which I am so grateful for.



My favorite hobby of all is threatening to tickle Chorin. I don't actually have to do it, just threaten, and he goes off as if I had done it. Cracks me up every time.

Why and when did you start using Linux?

I was introduced to Linux about 20 years ago, as a means to set up a web server. Redhat 5. I took to it like a fish to water, and did run it as a server for many years, but never considered it as a desktop system due to my being quite the gamer at the time ... until 2007, when I discovered PCLinuxOS. I've kept up with PCLinuxOS and other distros over the years, but was never ready to "take the plunge". I was very adept at Windows and all its deep inner-workings, and the thought of switching to something else where I'd basically be back to square one seemed pointless. Then along came Windows 10. That did it. I was ready to switch, and thankfully, I did. I can honestly say that I don't miss Windows one bit. It never even enters my mind. While the first month with PCLinuxOS was a major struggle, that came down to using Mate. Nothing against Mate, it just didn't seem to agree with my system somehow. Switching to KDE solved pretty much all of my problems.

What specific equipment do currently use with PCLinuxOS?

Toshiba Tecra R950 Laptop

Intel i7-3540M 3.0 ghz
 AMD Radeon 7500M/7600M
 HD Graphics 1 gb RAM
 16 GB RAM
 Realtek HD Audio
 500gb SSD
 PCLOS KDE 5 Plasma 2019

Laptop screen: 15" 1366x768
 External Monitor: 24" BENQ G2450HM 1920x1080

Do you feel that your use of Linux influences the reactions you receive from your computer peers or family? If so, how?

Absolutely. With the single exception of my wife, they all fall prey to the stigma that Linux is for geeks and gurus. They're basically afraid of it and too comfortable with Windows, in spite of its many issues. Short of taking my laptop everywhere and making people sit down and give PCLinuxOS a try (it worked with my wife!), there's just no convincing them.

What would you like to see happen within PCLinuxOS that would make it a better place. What are your feelings?

I don't feel that PCLinuxOS is lacking at all, nor needs anything to make it "better" (and DAMN, it boots up fast!). Obviously, Tex et al do an awesome job with this OS and I really can't ask for more. My laptop has never run faster or been more stable. As for the folks on the forum, I have a few things to say: First, it's so refreshing to visit a forum where there's a noticeable lack of in-fighting, power-struggles, and abuse. You're all grown-ups and you act like it. Everyone is so friendly and willing to share their time and knowledge on an ongoing basis. There's a palpable comradery here. I have the utmost respect for everyone on the forums, and I am very grateful for all your help and patience. And last, but certainly not least, you all put up with my special brand of crazy BS!

PCLinuxOS Family Member Spotlight is an exclusive, monthly column by YouCanToo, featuring PCLinuxOS forum member. This column will allow "the rest of us" to get to know our forum family members better, and will give those featured an opportunity to share their PCLinuxOS story with the rest of the world.

If you would like to be featured in PCLinuxOS Family Member Spotlight, please send a private message to youcantoo, parnote or Meemaw in the PCLinuxOS forum expressing your interest.

**PCLinuxOS
 Users Don't**

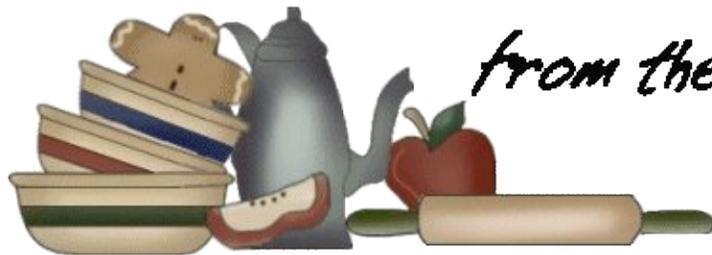
- Text
- Phone
- Web Surf
- Facebook
- Tweet
- Instagram
- Video
- Take Pictures
- Email
- Chat

While Driving.

**Put Down Your
 Phone & Arrive Alive.**



PCLinuxOS Bonus Recipe Corner



*from the kitchen of
youcantoo*

5-Ingredient Instant Pot™ Barbecue Pork Ribs

Ingredients

- 1 cup chicken broth
- 1 tablespoon barbecue seasoning
- 1 tablespoon Worcestershire sauce
- 3 lb baby back pork ribs,
cleaned and cut in 4 (3- to 4-rib) sections
- 1/2 cup sweet & spicy BBQ sauce

Directions

1. Spray 6-quart Instant Pot™ insert with cooking spray. Mix broth, seasoning and Worcestershire sauce in insert. Add ribs; turn to coat. Stand ribs up against sides of insert.
2. Secure lid; set pressure valve to SEALING. Select MANUAL; cook on high pressure 15 minutes. Select CANCEL. Keep pressure valve in SEALING position to release pressure naturally. Transfer ribs to cutting board; cool slightly. Discard cooking liquid. When cool enough to handle, cut sections into individual ribs.
3. Position oven rack 4 inches from broiler element. Set oven control to broil. Line rimmed sheet pan with foil. Spray with cooking spray. Place ribs on foil; brush all over with BBQ sauce.

4. Broil ribs 2 to 4 minutes or until sticky, crispy and blackened in spots.

Expert Tips

No broiler? Yes sir. No problem. While broiling adds nice flavor and color, the ribs will still be delicious if you need to skip that step.

Don't substitute larger ribs for the baby back ribs. This recipe was designed for the smaller racks, and larger ribs will not get tender in the same amount of time.



ms_meme's Nook: PCLOS Flyer



MP3

OGG

Won't you ride in my little red wagon
I'd love to pull you down Linux Street
All the Windows' kids will be jealous
When Texstar they see us meet

Hold tight when we come to the forum
Into the Sandbox we will coast
Won't you ride in my little red wagon
My PCLOS Flyer's the most

Won't you ride in my little red wagon
I'd love to pull you down Linux Street
For our OS we are so zealous
About it Texstar does tweet

Hold tight when we come to the forum
We will read every post
Won't you ride in my little red wagon
My PCLOS Flyer's the most

**International Community
PCLinuxOS Sites**



Like Us On Facebook!
The PCLinuxOS Magazine
PCLinuxOS Fan Club



Screenshot Showcase



Posted by tbschommer, May 25, 2019, running KDE.



PCLinuxOS Puzzled Partitions

2					8			
	3	8			4	5		
	7		1					
1				9				4
		3	5					
4	2							3
8				7		9		
		5					7	6
3						2		5

SUDOKU RULES: There is only one valid solution to each Sudoku puzzle. The only way the puzzle can be considered solved correctly is when all 81 boxes contain numbers and the other Sudoku rules have been followed.

When you start a game of Sudoku, some blocks will be prefilled for you. You cannot change these numbers in the course of the game.

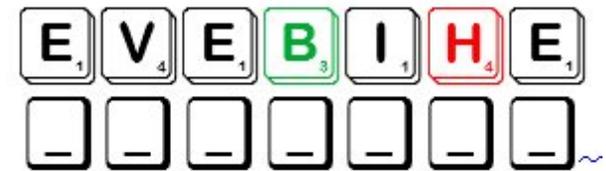
Each column must contain all of the numbers 1 through 9 and no two numbers in the same column of a Sudoku puzzle can be the same. Each row must contain all of the numbers 1 through 9 and no two numbers in the same row of a Sudoku puzzle can be the same.

Each block must contain all of the numbers 1 through 9 and no two numbers in the same block of a Sudoku puzzle can be the same.



SCRAPPLER RULES:

1. Follow the rules of Scrabble®. You can view them [here](#). You have seven (7) letter tiles with which to make as long of a word as you possibly can. Words are based on the English language. Non-English language words are NOT allowed.
2. Red letters are scored double points. Green letters are scored triple points.
3. Add up the score of all the letters that you used. Unused letters are not scored. For red or green letters, apply the multiplier when tallying up your score. Next, apply any additional scoring multipliers, such as double or triple word score.
4. An additional 50 points is added for using all seven (7) of your tiles in a set to make your word. You will not necessarily be able to use all seven (7) of the letters in your set to form a "legal" word.
5. In case you are having difficulty seeing the point value on the letter tiles, here is a list of how they are scored:
 - 0 points: 2 blank tiles
 - 1 point: E, A, I, O, N, R, T, L, S, U
 - 2 points: D, G
 - 3 points: B, C, M, P
 - 4 points: F, H, V, W, Y
 - 5 points: K
 - 8 points: J, X
 - 10 points: Q, Z
6. Optionally, a time limit of 60 minutes should apply to the game, averaging to 12 minutes per letter tile set.
7. Have fun! It's only a game!



Double Word



Triple Word



Download Puzzle Solutions Here

Possible score 275, average score 193.



PCLinuxOS Word Find: June 2019

Farming Terms

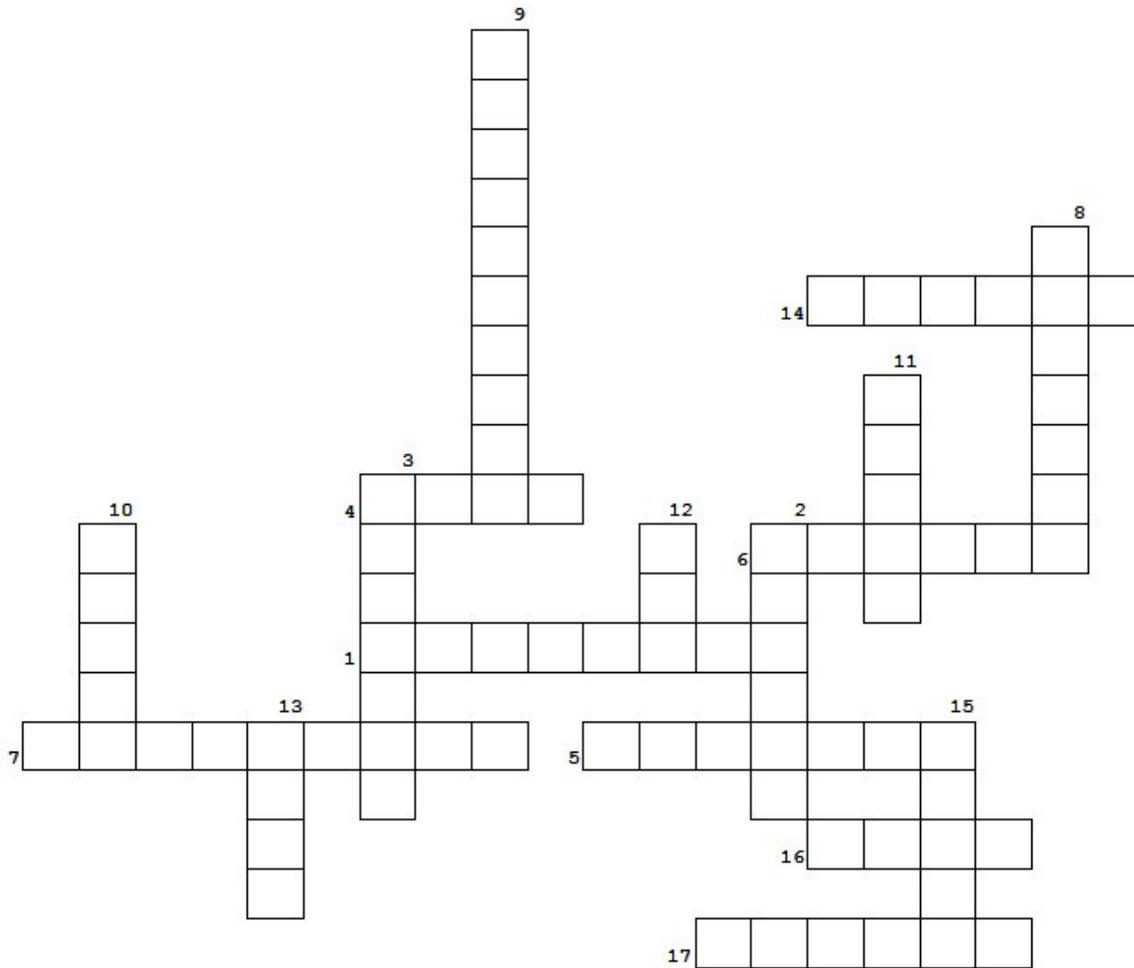
Z M A Z S O L X G V N J O X O L E V O H S V C K B X V R O Y
 D W M V F W F B N Q E X P U J L N J Q B H O U C R F R J Q X
 A O O D J W N M A N I G B O A R U W K R O T C A R T R U T U
 J H R L W H R O N G W D Q D G O C H P E O Z W J W N P R M F
 U I T R L E L Y S I N C U B A T O R W K E T N Z E O B Q O V
 A I S E R A M L M I P W T L R A H J Y Q R E W Z V I S K Q I
 Q A G Z F T F W N F B O Q G X V O K L N S L N E I T H O I K
 L A O F K J E X A Z Z D H A Z I M Z B S Y L E H H A J E K R
 X B U L L M Q R W E V A J M P T S U K X C U L X E G N X L V
 V B S J Y V M R N U F E R T I L I Z E R T P C L E I W K D N
 X Q C D J E N S T N D M N B O U V R D X R P T M B R P Z D K
 S P I E R U T L U C I R G A R C M A C H E T E M K R O I D F
 Y F D Q W T V L L Y L S K F C O G Y R I A D O G U I K D U N
 H E D D O N K E Y Z F S Q Z H F C H D C H C E Q Q Y T A U C
 L C A G H Z X L J U A P I J A Z H J D D S E K O O M J R U N
 Z S L W R O T K H A L B D U R N L I O Y Q Q S C Y T H E K M
 Q P C G B B A C U J F S S U D M H V B B V X O R K E V Z J R
 B N P B B B K I M M Z G S E U D L N D E W P U M Q R B Y I S
 Y C R Z K I O S B A K I V W T H E D B R M J U A Q E E J Y I
 X Z Y R T L T R E R D Q S O A S J R X A A L W R C I V W W E
 Q W D U T C V I N B W U Z T J Z E P M C C K B W Q U R D U W
 U K V A B N G F L X G N C P A D Q V N H O N E Y B E E W J C
 B G T A K R E D N L Y H O G A O K Q R C H I C K E N S W H O
 C D O L J O D K M P E H E O V J N P O A M A L L O W I G E X
 O Q X Z L H X Q I R A R R T W U P M W Q H N S L E N E G S U
 D M L I Y G C O Y Z S F E V U R I X I R Y S F J D J N J P X
 M H S A W N K T J A J K L D G R K N N R X H I M K U R G M X
 H U W W D O M Z V M C R A T X K M A E J K Z I N I F S O Z K
 D P B M K L Z W X U J W B I M P F C X J D L G B R G G Z V E
 U K D N I R O B B J A Z I P O W G J K J L J J E D R B E R U

- | | |
|-------------|------------|
| agriculture | baler |
| beehive | bison |
| boar | bucket |
| buffalo | bull |
| cattle | chicken |
| combine | cultivator |
| dairy | donkey |
| drake | ewe |
| fallow | farmer |
| fertilizer | foal |
| harvest | hatchery |
| honeybee | incubator |
| irrigation | jack |
| jenny | llama |
| longhorn | machete |
| mare | meadow |
| mulch | oats |
| orchard | pullet |
| scythe | shovel |
| sickle | silo |
| tiller | tractor |
| udder | wheat |
| windmill | |

[Download Puzzle Solutions Here](#)



Farming Terms Crossword



1. facility where eggs are hatched under artificial conditions, especially those of fish or poultry.
2. a tool with a long curved blade at the end of a long pole attached to which are one or two short handles.
3. a broad-bladed weapon used as an ax or a sword
4. adult female horse
5. grove of fruit trees
6. a short-handled farming tool with a semicircular blade
7. machine which maintains a warmer temperature to care for babies
8. machine that harvests farm crops
9. machine that tills the soil
10. buffalo
11. a layer of material applied to the soil, most often to help curb weed growth, or for decoration
12. female sheep
13. a wild pig or a male hog
14. crop ground left unplanted in crop rotation
15. male duck
16. baby horse
17. a hen less than a year old

[Download Puzzle Solutions Here](#)

Mixed-Up-Meme Scrambler



If we've told you once, we've told you a million times.

Don't Open the

— — — — —

Without Permission!!

Dirt
THILF

— — — — —

Agent
IFXER

— — — — —

Thread
IBFER

— — — — —

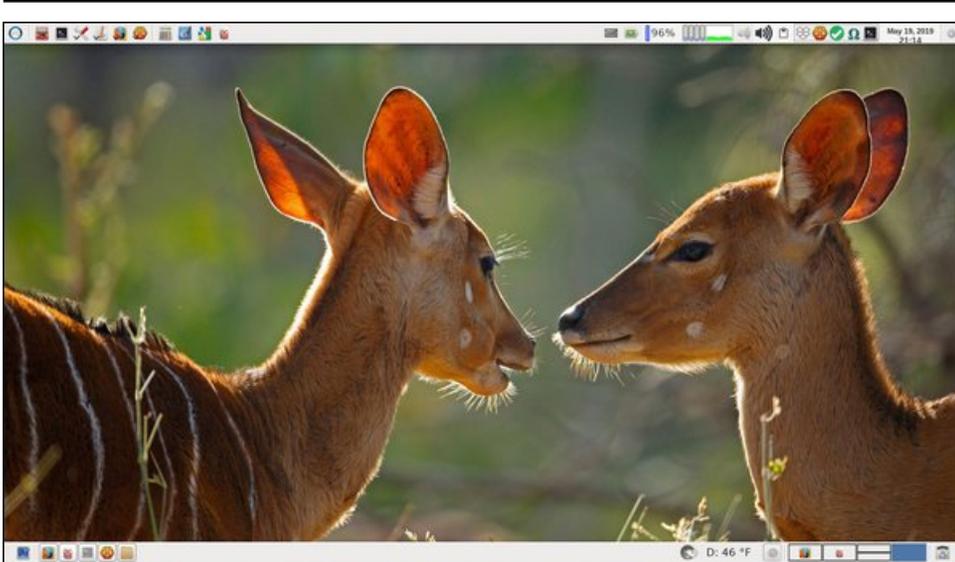
Shut
LCOSE

— — —

Use the clues to unmix the letters to make a new word. Remix the letters in the red boxes to solve the puzzle.

[Download Puzzle Solutions Here](#)

More Screenshot Showcase



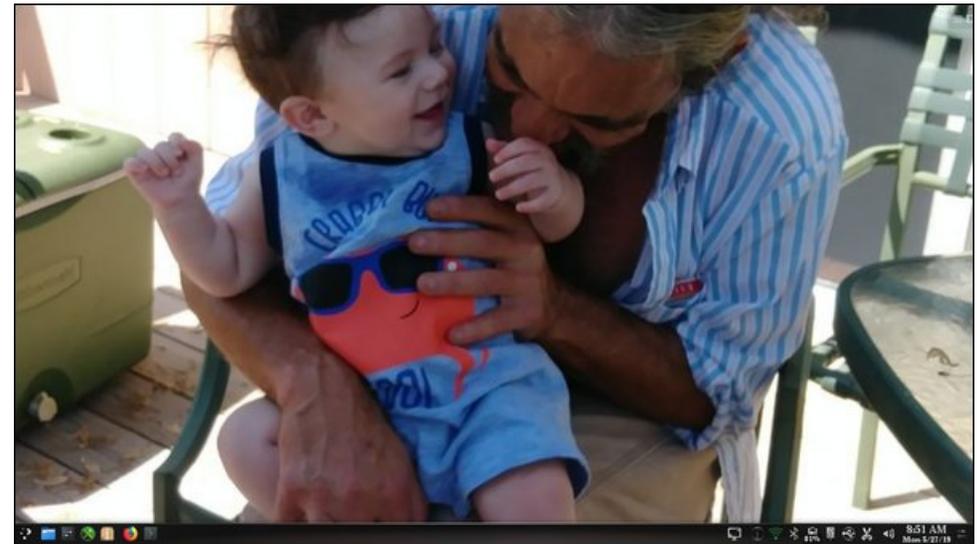
Posted by parnote, May 19, 2019, running Xfce.



Posted by tuxlink, May 2, 2019, running KDE.



Posted by Meemaw, May 20, 2019, running Xfce.



Posted by ShowMeRon, May 27, 2019, running KDE.