

Costruire un segreto

Perché estrarre nomi da un cappello quando uno script può farlo al posto vostro? Linux Pro vi mostrerà come produrre questo script in PHP CLI



Con le feste appena passate, appesantiti da panettone, torrone e cenoni, perché non rimettersi in forma programmando?

Un ottimo spunto per uno script utile può venire dal Natale appena passato. Chi non ha mai partecipato a un "Babbo Natale segreto"? Questa usanza è diffusa in tutto il mondo con diversi nomi, tra i quali, Pollyanna, Kris Kringle o appunto Secret Santa; per chi non la conoscesse consiste nello scrivere i nomi di ogni partecipante su di un pezzo di carta, inserirli tutti in un cappello e poi ognuno estrae un nome dal cappello. Il nome della persona estratta sarà la persona a cui si dovrà fare un regalo. Spesso per rendere le cose più eccitanti il nome rimane segreto (ecco perché "Secret") fino al momento dello scambio dei regali. A volte il gruppo opta per complicare un po' le cose aggiungendo altre regole. Ad esempio, in un gruppo di parenti si può vietare di fare regali a persone del proprio nucleo familiare stretto. Un'altra regola spesso utilizzata è

che non si può pescare la stessa persona pescata l'anno precedente. Ad esempio, quando il cappello viene passato a Massimiliano, lui pesca il nome della fidanzata o della persona a cui ha fatto il regalo l'anno prima, lo rimette nel cappello e ne pesca un altro. Se il giro finisce in stallo, cioè quando l'ultima persona che deve pescare può pescare soltanto una persona "non valida", si può modificare il risultato manualmente oppure rimettere tutti i nomi nel cappello e ricominciare con l'estrazione.

Scriptare con PHP CLI

Piuttosto di utilizzare l'obsoleto, seppur affascinante, metodo del cappello, vi mostreremo come scrivere uno script che si occupi di automatizzare l'intero processo, assicurando correttezza nelle estrazioni e permettendovi di gestire con maggior semplicità il problema dello stallo. Per semplicità, vi faremo scrivere questo script in PHP CLI, una versione da riga di comando del famosissimo linguaggio di scripting Web PHP di cui avete appreso le basi il mese scorso. Per installare PHP CLI in Ubuntu, dovete eseguire il seguente comando:

```
sudo apt-get install php5-cli
```

Ora provate a scrivere un semplice script per verificare che tutto sia andato a buon fine. Per partire, create un file di test per contenere il vostro codice e rendetelo eseguibile:

```
cd ~
touch ciao.php
chmod +x ciao.php
```

A questo punto aprite il file **ciao.php** con il vostro editor di testo preferito e inserite quanto segue:

```
#!/usr/bin/php -q
<?php
echo "Ciao Mondo!"
?>
```

Per testare lo script è sufficiente digitare **./ciao.php** dal prompt della riga di comando, e dovrebbe apparire il messaggio **Ciao Mondo!** nel vostro terminale.

Costruire un cappello virtuale

Esattamente come fatto per lo script di prova, iniziate creando un file che andrà a contenere il codice PHP e rendete il file eseguibile:

```
cd ~
touch natale.php
chmod +x natale.php
```

Ora aprite **natale.php** con un editor e inserite il codice seguente (la numerazione a inizio riga non è da digitare, è stata inserita solo a scopo esemplificativo):

```
01 #!/usr/bin/php -q
02 <?php
03 passthru('clear');
04 $persone = array('Persona 01',
05                  'Persona 02',
06                  'Persona 03',
07                  'Persona 04',
08                  'Persona 05',
09                  'Persona 06',
10                  'Persona 07',
11                  'Persona 08',
12                  'Persona 09',
13                  'Persona 10',
14                  'Persona 11',
15                  'Persona 12',
16                  'Persona 13',
17                  'Persona 14',
18                  'Persona 15');
```

La prima riga di questo codice gli permette di essere eseguito da riga di comando. Quando la nostra shell vede questa dicitura, sa che deve utilizzare PHP CLI come interprete. La seconda riga è il tag di apertura del PHP e la terza riga dice alla shell di pulire la finestra del terminale. Le righe dalla 4 alla 18 definiscono un array che contiene i nomi di tutti i partecipanti al Babbo Natale segreto. I nomi scritti in ogni elemento dell'array rappresentano i bigliettini su cui si scrivevano i nomi da mettere nel cappello. Ora dovete creare un cappello virtuale, per contenere i foglietti con i nomi dei partecipanti. Fatelo creando una copia del vostro array **Persone**.

19 \$foglietti = \$persone;
Ora avete un array con i partecipanti e un array che funge da cappello virtuale, contenendo i foglietti con i nomi di ogni partecipante. A questo punto dovete scrivere un po' di codice che estragga un nome dal cappello per ogni partecipante. Per fare questo, aggiungete le seguenti righe di codice alla fine del vostro file:

```
20 for ($riga = 0; $riga <= count($persone)-1; $riga++) {
21     $cellaCasuale = array_rand($foglietti);
22     echo $persone[$riga]. "t--> \n";
23     unset($foglietti[$cellaCasuale]);
24 }
25 ?>
```

La linea 20 è un ciclo **for** che scorrerà l'array **persone**, l'equivalente di far girare il cappello. La linea 21 sceglierà un nome a caso dal nostro array **foglietti**. La linea 22 mostra il risultato all'utente, mentre la 23 elimina dall'array il nome appena estratto in modo che non venga ripescato. La linea 24 è semplicemente il tag di chiusura del ciclo **for**, la 25 contiene la chiusura del tag **php**. A questo punto salvate lo script ed eseguitelo dal prompt dei comandi per vedere la lista dei partecipanti con a fianco la loro scelta casuale. Eseguite il codice un po' di volte e guardate bene le coppie di nomi. Notate niente? Se siete osservatori particolarmente acuti potreste notare che alcuni nomi sono in coppia con loro stessi, in quanto non c'è nessun controllo per evitare che questo accada. Nella realtà questo problema è risolto semplicemente rimettendo il nome nel cappello e pescandone un altro.

Rifinire il codice

Aggiungere una parte logica per rimettere le pescate sbagliate nel cappello è abbastanza lineare. Aprite **natale.php** e sostituite le righe 22-23 con le sette righe di codice qui sotto:

```
22     if ($persone[$riga] == $foglietti[$cellaCasuale]) {
23         if (count($foglietti) <= 1) { echo "\n Stallo in atto ". $persone[$riga]. "\n\n"; exit; }
24         $riga--;
25     } else {
26         echo $persone[$riga]. "t--> \n";
27         unset($foglietti[$cellaCasuale]);
28     }
```

La riga 22 ora contiene il confronto tra il foglietto selezionato e il nome della persona che sta facendo la pescata. Se il partecipante non pesca se stesso, la linea 26 mostra il risultato e la 27 rimuove la pescata appena effettuata dal cappello in modo che non possa essere ripescata, esattamente come prima. Se invece pesca se stesso, la linea 24 riporta indietro il contatore in modo che sembri che la pescata non sia mai accaduta, e lo script procederà a pescare di nuovo. Abbiamo tenuto la riga 23 per ultima perché è una parte molto importante del codice. Se non inserite questa riga può capitare, in rari casi, che il programma rimanga appeso in un loop infinito. Questo succede quando l'ultima

Un esercizio sulle probabilità

In uno scenario di Babbo Natale Segreto tradizionale, ogni persona deve pescare il nome di un'altra persona, in altre parole un qualunque nome tranne il proprio. In matematica, una permutazione di questo tipo è chiamata *dismutazione*. In parole povere, una dismutazione è una permutazione di un insieme di elementi tale per cui nessun elemento appare nella posizione originaria. Le probabilità di raggiungere una dismutazione di successo da un tipico scenario di Babbo Natale Segreto può essere calcolato matematicamente. Comunque, invece di tentare di calcolare questa probabilità tramite la matematica, potete stabilirla attraverso una simulazione al computer, realizzata eseguendo lo script di questo tutorial migliaia di volte e facendo la media dei risultati. Nella simulazione con 10, 20, 30, 40 e 50 persone, eseguendo lo script 10.000 volte (metodo Monte Carlo), siamo arrivati a questi risultati:

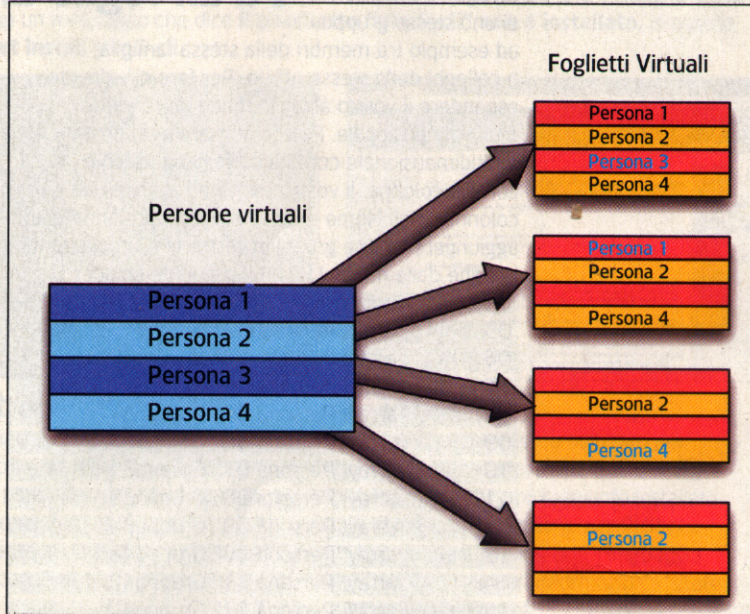
Partecipanti	Probabilità
10	64,15%
20	63,08%
30	62,86%
40	63,04%
50	64,19%

persona che pesca l'ultimo foglietto dal cappello si ritrovi con il foglietto con il proprio nome. Potete rimetterlo nel cappello e ripescarlo quante volte volete, ma il risultato sarà sempre lo stesso, questo è un loop infinito. Il codice alla riga 23 testa questa condizione, conosciuta come *stallo*, e se si avvera, informa l'utente che è avvenuto uno stallo e termina l'esecuzione dello script.

Mescolare il tutto

Eseguendo il vostro script, noterete che i partecipanti pescano sempre nello stesso ordine. Dato che lo script pesca i foglietti casualmente, l'ordine di estrazione non è molto importante, ma per


Una rappresentazione grafica dei partecipanti virtuali e dei foglietti virtuali



L'output di questo script mostra un caso di stallo

```
ninjak@ninjak-HP-Pavilion-dv7-Notebook-PC: ~/Dropbox/LXP113/rehardcorelinuxphp
Persona 12 --> Persona 03
Persona 15 --> Persona 04
Persona 10 --> Persona 15
Persona 03 --> Persona 13
Persona 05 --> Persona 12
Persona 07 --> Persona 01
Persona 02 --> Persona 05
Persona 14 --> Persona 10
Persona 04 --> Persona 14
Persona 01 --> Persona 06
Persona 09 --> Persona 02
Persona 11 --> Persona 08
Persona 06 --> Persona 11

Stallo in atto Persona 08!

ninjak@ninjak-HP-Pavilion-dv7-Notebook-PC:~/Dropbox/LXP113/rehardcorelinuxphp$
```

rendere le cose un po' più realistiche, vi suggeriamo di aggiungere una riga di codice per rimescolare l'array **persone** prima di iniziare a pescare.

```
20 shuffle($persone);
```

Aggiungere questo brano di codice alla riga 20 ha l'effetto di rendere l'ordine dei partecipanti casuale come l'ordine dei foglietti.

Un affare di famiglia

A questo punto avete uno script di "Babbo Natale segreto"

perfettamente funzionante, ma supponete di voler evitare lo scambio di regali tra membri di uno stesso gruppo,

ad esempio tra membri della stessa famiglia o colleghi dello stesso ufficio. Per fare ciò, dovete espandere il vostro array in modo da renderlo multi-dimensionale. Potete pensare a un array multidimensionale come a un foglio di calcolo con più colonne. Il vostro nuovo array avrà una colonna per il Nome e una per il Gruppo. Andate ad aggiungere cinque gruppi al vostro test rimpiazzando le righe dalla 4 alla 18 con il seguente codice:

```
04 $persone = array(array('Persona 01','Gruppo1'),
05 array('Persona 02','Gruppo1'),
06 array('Persona 03','Gruppo1'),
07 array('Persona 04','Gruppo2'),
08 array('Persona 05','Gruppo2'),
09 array('Persona 06','Gruppo2'),
10 array('Persona 07','Gruppo3'),
11 array('Persona 08','Gruppo3'),
12 array('Persona 09','Gruppo3'),
13 array('Persona 10','Gruppo4'),
14 array('Persona 11','Gruppo4'),
15 array('Persona 12','Gruppo4'),
```

```
16 array('Persona 13','Gruppo5'),
17 array('Persona 14','Gruppo5'),
18 array('Persona 15','Gruppo5'));
```

Se provate a eseguire lo script a questo punto noterete che funziona senza errori, ma invece di mostrare i partecipanti e le loro pescate, mostra solo la parola **array**. Questo è un effetto secondario dell'aver reso multidimensionale l'array. Per rendere pienamente funzionante lo script, come faceva prima della modifica, dovete essere un po' più specifici

su cosa volete si veda. In questo caso dovete aggiungere **[0]** vicino a ogni parentesi quadra, tranne alla riga 28 (l'unset deve

"Potete pensare a un array multidimensionale come a un foglio di calcolo"

rimanere com'è) - guardate il codice della riga 23 riportato sotto per capire come modificare il file - così PHP saprà che volete visualizzare la colonna Nome o, più correttamente, l'elemento zero di ogni

Molti percorsi per lo stesso risultato

Se siete novizi della programmazione, è facile che vi ritroviate in difficoltà quando dovete raggiungere un particolare risultato. Se lo scripting è il vostro pane quotidiano, invece, saprete di certo che raramente esiste un'unica risposta giusta a una richiesta. Questo dilemma si presenta spesso prima che un programmatore scriva la prima riga di codice. Il PHP vi offre la possibilità di adottare un approccio object-oriented (OO) o un approccio procedurale. Come per quanto riguarda il dibattito tra Vi ed EMACS, molta gente sceglie una strada e ritiene che l'altra sia scorretta. Non fatevi coinvolgere da queste diatribe. Scegliete l'approccio più funzionale per voi, lavorateci e divertitevi. Non c'è dubbio che ci siano vantaggi e svantaggi in ogni approccio, ma nessuna strada è giusta o sbagliata.

riga del vostro array. Eseguite ancora il codice e vedrete che il programma funziona di nuovo come prima di modificare l'array. A questo punto dovete dire allo script di utilizzare i dati sui gruppi appena aggiunti. Per fare ciò cambiate la riga 23 come segue:

```
23 if (($persone[$riga][0] ==
    $foglietti[$cellaCasuale][0]) || ($persone[$riga][1]
    == $foglietti[$cellaCasuale][1])) {
```

Quello che avete fatto ora è dire a PHP di non controllare soltanto di non aver pescato lo stesso nome, ma anche di non aver pescato il nome di qualcuno nel nostro stesso gruppo. Le due barre verticali, chiamate pipe, dicono a PHP di valutare l'espressione alla loro sinistra e alla loro destra, e se almeno una delle due è vera esegue le azioni all'interno delle parentesi graffe. Questo operatore è chiamato **or**, ed è estremamente utile per scrivere del codice compatto e pulito; in realtà non sarebbe più necessario testare la prima parte dell'or perché se è uguale la persona è sicuramente uguale anche il gruppo a cui appartiene, ma abbiamo preferito lasciarla per scopi didattici. Se lanciate il vostro script ora noterete che funziona a volte, ma altre volte non termina e rimane appeso in un loop infinito. Il problema si presenta perché ogni volta che viene pescato un nome dal cappello è il nome di una persona nello stesso gruppo di chi sta pescando. Esistono diversi modi per aggirare questo problema. In questo caso, aggiungerete un semplice meccanismo che porta il numero di tentativi possibili a 100. Se non riuscirete a raggiungere l'obiettivo di accoppiare tutti in 100 tentativi, getterete

la spugna dichiarando lo stallo e uscirete dallo script. Per raggiungere questo traguardo dovete aggiungere una riga prima del ciclo for ed effettuare due piccole modifiche alle righe 25 e 26 (o 24 e 25 prima di aggiungere la nuova riga).

```
21 $tentativi = 0;
25 if ((count($foglietti) <= 1) || ($tentativi >=
    100)) {echo "\n Stallo in atto
    ". $persone[$riga]. "\n\n"; exit; }
26 $riga--; $tentativi++;
```

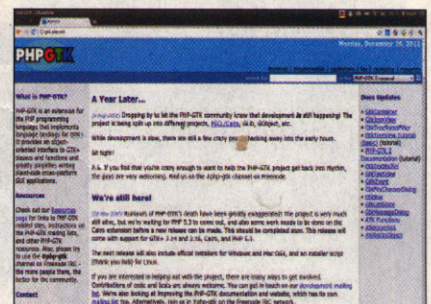
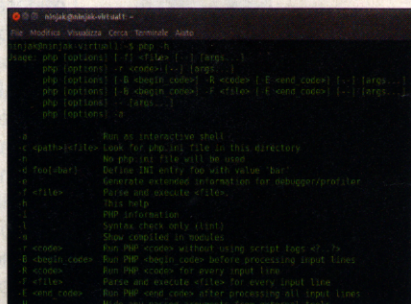
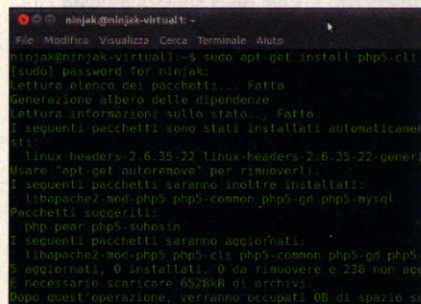
Alla riga 21 create una variabile chiamata **tentativi**, alla riga 26 la incrementate di uno ogni volta che viene superato il controllo alla riga 24. Questo ha l'effetto di rimettere il foglietto nel cappello e ripescare. Se dovesse succedere per 100 volte, la linea 25 provvederà a informare l'utente che si è verificata una situazione di stallo e lo

"L'operatore or è estremamente utile per scrivere del codice compatto e pulito"

script verrà terminato. Se lanciate il vostro script ora, vi ritornerà un set completo di pescate, che rispetterà i requisiti di accoppiare chi fa il regalo e chi lo riceve in base al gruppo. A seconda del numero di partecipanti e del numero di gruppi, potreste vedere più o meno frequentemente lo stallo. Se succede, eseguite nuovamente lo script fino a ottenere un risultato completo. Per migliorare la vostra abilità di programmare, aggiungete dei controlli per evitare che le persone facciano regali alle persone a cui li hanno fatti l'anno prima o per far ripartire lo script in caso di stallo. Un altro passo potrebbe essere quello di permettere al vostro script di accettare i partecipanti e di visualizzarli tramite il browser Web. Provateci e condividete con noi i vostri risultati! **LXP**

Testare la vostra installazione PHP

Il modo più semplice di testare se PHP CLI è già installato nel vostro sistema è di aprire il terminale e scrivere **php** sulla riga di comando. Un'altra opzione è di scrivere **sudo dpkg -s php5-cli**. Se PHP CLI è installato correttamente, vedrete un messaggio che include la dicitura **status: install ok installed**. Se non è installato riceverete un messaggio che dice il pacchetto **"php5-cli" non è installato**, in questo caso installarlo è una banalità, basta digitare **sudo apt-get install php5-cli**.



1 Dite semplicemente sì

Installate php5-cli utilizzando il comando **sudo apt-get install php5-cli** quando apt vi chiede se volete continuare o meno, rispondete di **Sì** e il gioco è fatto.

2 Controllate la versione

Quando l'installazione è terminata, potete testarla scrivendo **php -v**, e PHP CLI dovrebbe rispondervi con le informazioni sulla versione. Per saperne di più relativamente ai parametri che potete utilizzare con PHP CLI, scrivete **php -h**.

3 PHP-GTK

PHP può anche essere utilizzato per costruire applicazioni desktop. Questo obiettivo può essere raggiunto installato PHP-GTK, un'estensione per PHP che implementa il binding con il linguaggio GTK+. Per maggiori informazioni visitate <http://gtk.php.net>.