Using The Listgen Functions in your GNU/Linux Shell Scripts

Add the power of interactive menus, buttons and multi-column lists

Elizabeth Audrey Mills

Using The Listgen Functions
in your
GNU/Linux Shell Scripts

by

Elizabeth Audrey Mills

Using The Listgen Functions in Your GNU/Linux Shell Scripts

Published by Elizabeth Audrey Mills

~~~~~

Free Edition

First revision 9th January 2017

~~~~~

~~~~~

Written using LibreOffice by The Document Foundation

~~~~~

~~~~~

Email: elizabeth@itsliz.net

Website: www.itsliz.net
~~~~~

Elizabeth Audrey Mills is the author of the Natalie Tereshchenko series of historical novels, and the Tapestry Capricorn science-fiction novelettes for older children. In her spare time, she taught herself shell scripting, and wrote the independent Feliz scripts for installing Arch Linux. Feliz stands out as being the quickest and easiest Arch installer, with many enthusiastic reviews on YouTube. The scripts are open-source, and freely available through Github …

https://github.com/angeltoast/feliz2

… and as a live iso installer from Sourceforge …

https://sourceforge.net/projects/feliz/

Using The Listgen Functions
in your
GNU/Linux Shell Scripts

Introduction

The listgen scripts are part of the independent Feliz installer for the Arch Linux operating system. They are written entirely in bash, so you can edit them to suit your purposes.

In this book, aimed at intermediate scripters, we will look at how the functions in the listgen module can be incorporated to help you to improve the display of information while running your own scripts.

The listgen module is self-contained; it initialises all its own variables, and contains all the functions it needs. You only have to put a copy of the module in the same location as your scripts, and source it, to be able to add all the features of listgen to your scripts.

Shell scripting fits right into the classic UNIX philosophy of breaking complex projects into simpler sub-tasks, of chaining together components and utilities – what I like to call 'tools'. Many consider that this a better - or at least more efficient - approach to problem-solving than using one of the higher-level languages.

This is not a book for beginners. If you have never scripted before, it is recommended that you first learn a few of the basics from such websites as …

http://linuxcommand.org/lc3_writing_shell_scripts.php

The examples in this book come from the development of the Feliz set of scripts for installing the Arch Linux operating system. The entire installer is written in bash, using only the standard system tools that ship with the Arch Linux live session. If you want to see how the techniques in this book are built together into a full set of working scripts, you can see them all at …

https://github.com/angeltoast/feliz2

GNU/Linux basics

The system known throughout the world today as 'Linux' grew from Unix, a commercial operating system that was developed at the AT&T Bell Labs in the 1970s by Ken Thompson, Dennis Ritchie, and others. GNU/Linux was born when Richard Stallman began the GNU project in 1984, with the aim of creating a free version of Unix. He had a vision of software that could be freely used, read, modified, and redistributed, and his Free Software Foundation built most of the components that are now part of GNU/Linux, essentially an entire operating system that lacked only a working kernel to be fully functional. This final need was met in 1991, when Linus Torvalds developed a kernel, which he named Linux.

GNU/Linux follows the Unix philosophy that the operating system should provide a unified filesystem, a set of tools that each performs a limited, well-defined function, and a shell scripting and command language to combine the tools to perform complex workflows.

And, yes, I am going to be a real pain and use the full term 'GNU/Linux' throughout this book, because I believe it is important.

A Summary of The listgen functions

The listgen functions have been an integral part of the Feliz installer since its birth, and have grown with it. They are designed to help with creating intuitive cli interfaces without using Dialog, Whiptail, or any other external package. Instead, the tput library is used to determine the terminal dimensions and to display lists and text centrally according to content.

The listgen script is designed to be self-contained and application-independent, and any of the functions may be called independently from any scripts, as long as the listgen script is available and sourced, or the functions are copied into your script.

The listgen module is self-contained. The functions are:

Heading - Prints the contents of a variable $Backtitle (which may be set elsewhere in your scripts) in a banner at the centre-top of the terminal or screen.

first_item – Used by the list-printing functions to centralise the first item of a menu;

subsequent_item – Used by the list-printing functions to align successive menu items;

PrintRev – Reverses (ie: highlights) text colour for selected menu item and/or button;

Buttons - Prints a row of buttons;

ActiveMenu - Controls the selection and highlighting of menu items;

listgen1 - Generates a menu of one-word items;

listgen2 - Generates a menu of multi-word items;

listgenx - Generates pages of numbered lists, for lists that would exceed screen size;

SelectPage - Selects appropriate page according to user input;

PrintPage – Displays the selected page of data, fitting as many columns as possible according to the terminal/screen size.

Getting listgen

To use the functions in the listgen module, download the latest copy from my Github repository.

First open a terminal in the directory where you want to store it, then type ...

wget https://raw.githubusercontent.com/angeltoast/feliz2/master/listgen

In the main module of your scripts, add listgen to your sources …

source listgen

… or …

. listgen

A simple example, using the 'Buttons' function

Buttons can be used to generate an interactive Yes/No display.

A call to buttons uses three arguments:

1) Type (Menu, Yes/No);

2) Button string. This should contain one or two words: eg: 'Ok' or 'Ok Exit'

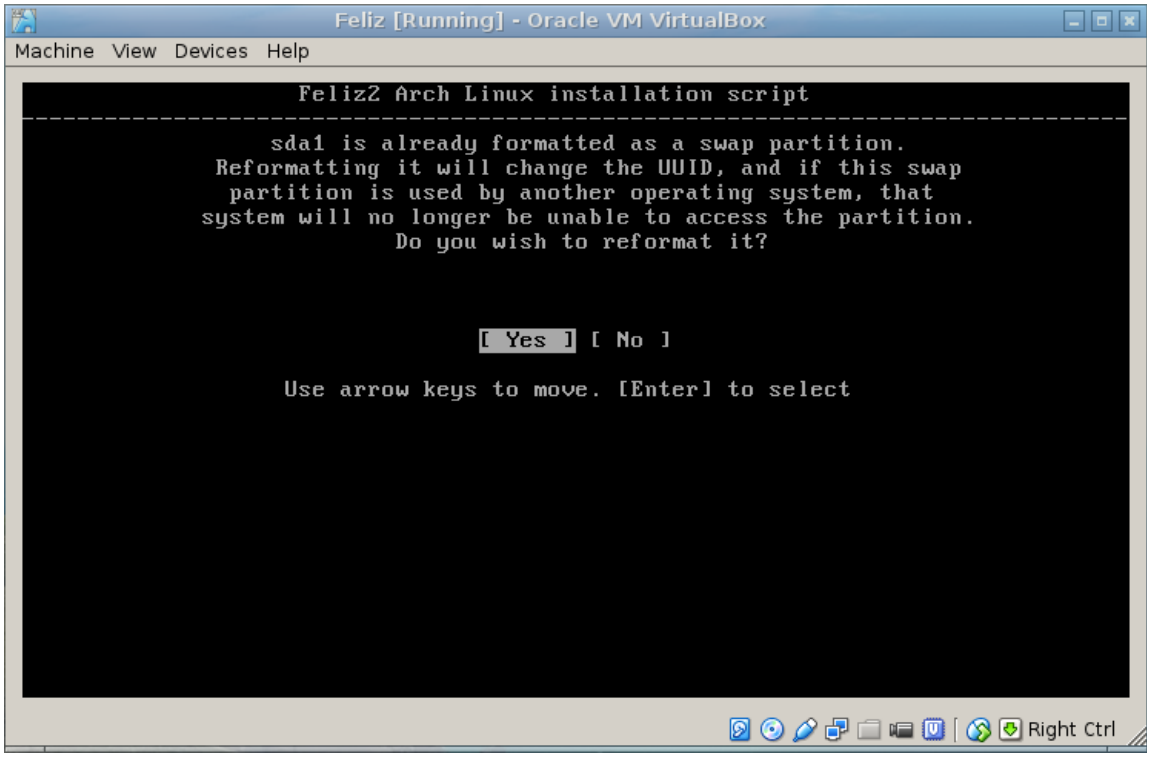
3) Message string.

At the time of writing, Buttons is limited to displaying only two buttons, but this may be extended to three options in the near future.

Call Buttons with a line like this:

Buttons "Yes/No" "Yes No" "Use arrow keys to move. [Enter] to select"

The arguments in the example above inform the Buttons function of the type of operation, that the the buttons are to be labelled 'Yes' and 'No', and a short line of guidance to the user is to be displayed.



listgen1

The first and simplest listing function, listgen1, is for use in shell scripts to generate an interactive menu. It prints the list of options provided, in the form of a scrollable menu, with a pair of 'buttons' to confirm or exit.

Arguments:

1) A simple string array variable of the items to be listed;

2) An optional one-line string of text to be displayed beneath the 'buttons', advising the user how to use the menu. If this argument is passed as an empty string, the menu will display without the instructions line. This is for situations where a menu may be close to filling the screen, or where no instructions are necessary;

3) Button text eg: 'Ok Exit' or just 'Ok'. These must be single-word options. If only one word is passed (eg: 'Ok') then no exit option will be printed; in this way you can make sure that your users pick one of the menu items offered.

A call to listgen1 takes the form ...

listgen1 "argument1" "argument2" "argument3"

All three arguments must be passed, although the second argument can be an empty string, if no message is required.

Example call:

listgen1 "Reboot Shutdown" "" "Ok Exit"

The first (string array) argument may be passed in one of three ways:

1) It may be specified in the calling line (as above):

listgen1 "item1 item2 item3 ... "

2) It may be assigned to a variable:

VariableName="item1 item2 item3 ... "

For example ...

Accessories="Conky Geany Nautilus Terminator"

listgen1 "$Accessories"

3) It may be generated by a bash command ...

For example …

Partitionlist=$(lsblk -l | grep 'part')

listgen1 "$Partitionlist" "" "Ok"

Note: A bash-generated list should be converted to a string array. For example:

```
Zones=(`timedatectl list-timezones | sed 's/\/.*$//' | uniq`)
passzones=""
for z in ${Zones[@]}; do
   passzones="$passzones $z"
done
 ...
listgen1 "${passzones}" "" "Ok"
```

listgen1 sets two global variables for use by the calling function ...

1) $Response - the item number selected by the user, and

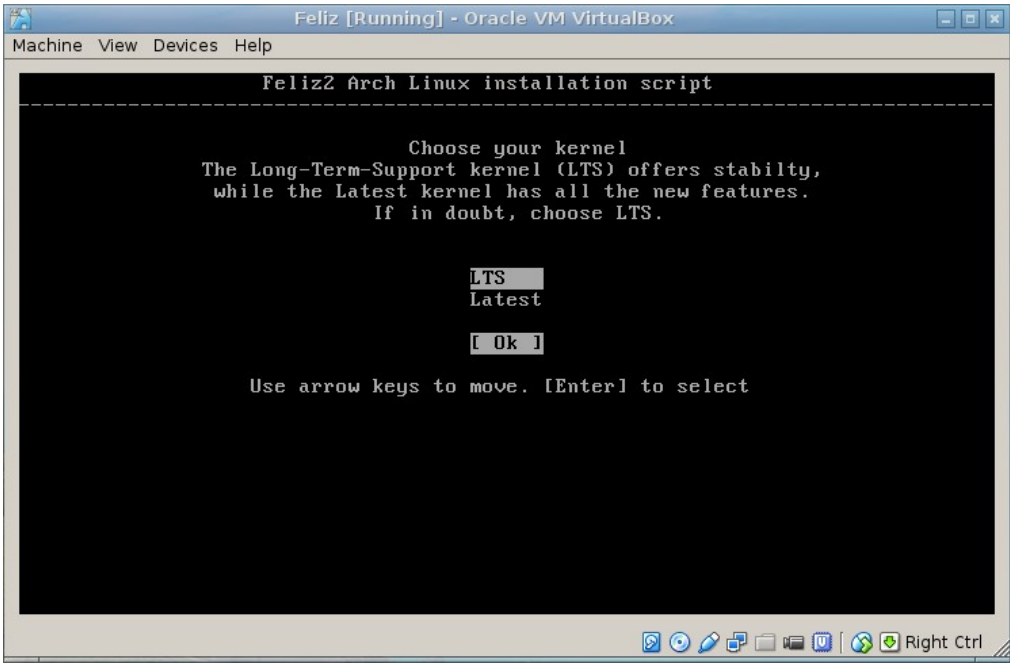
2) $Result - the label (from the variable array) of the item selected.

This means that you can respond to the user input by using (for example) a 'case - esac' statement or an 'if – else- fi' statement.

Examples from Feliz

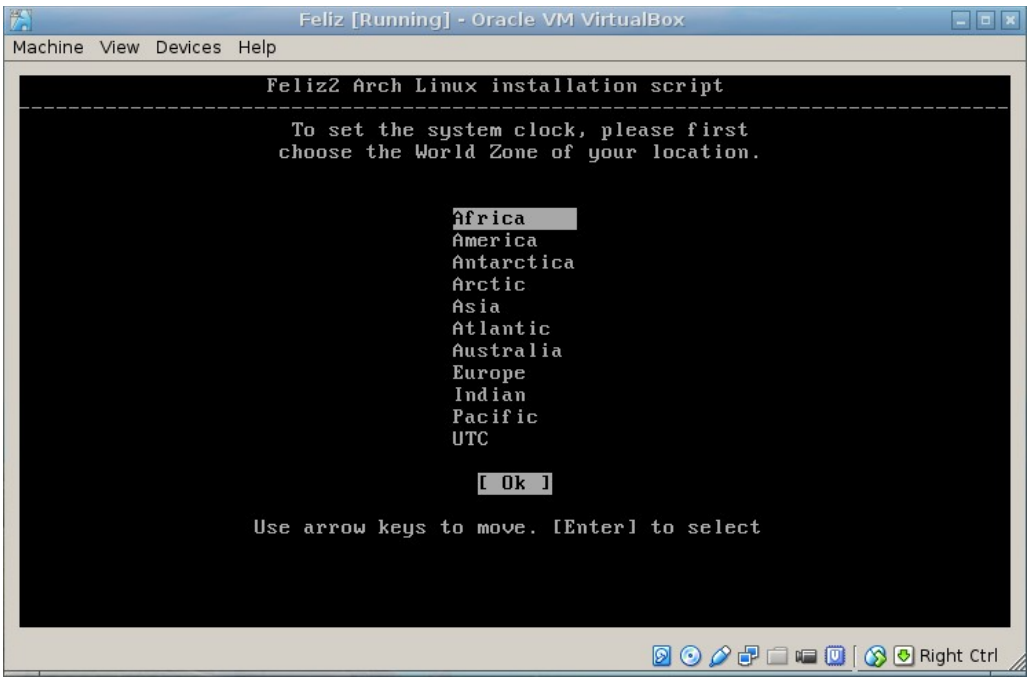
Example (1) A simple, two-options menu:

```
listgen1 "LTS Latest" "Use arrow keys to move. [Enter] to select" "Ok"
#  Listgen will return 1 for the first item (LTS)
#  or 2 for the second item (Latest) via the $Response variable
#  You can then use this for further processing
Kernel=${Response}          # Set the Kernel variable
```



Example (2) Displays the Ten world zones

```
Zones=$(timedatectl list-timezones | cut -d'/' -f1 | uniq) # Ten world zones
echo
zones=""
for x in ${Zones[@]}                    # Convert to space-separated list
do
    zones="$zones $x"
done
listgen1 "${zones}" "Use arrow keys to move. [Enter] to select" "Ok"
ZONE=$Result
```


Feliz [Running] - Oracle VM VirtualBox
Machine View Devices Help
Feliz2 Arch Linux installation script
To set the system clock, please first
choose the World Zone of your location.
Africa
America
Antarctica
Arctic
Asia
Atlantic
Australia
Europe
Indian
Pacific
UTC
[Ok]
Use arrow keys to move. [Enter] to select
Right Ctrl

listgen2

listgen2 is designed for displaying longer descriptions in a menu. It receives the text of the long items via an array. When an item is selected, listgen2 finds that item in a reference array of short (one-word) names that match the array of long-descriptions, and saves it as the global variable $Result which may then be used by the calling function if desired. Although complex to set up, it adds greater functionality and user-friendliness for menus. The reference array must be specifically declared, and the elements filled, before it can be used.

Calling listgen2 takes the form:

listgen2 "$PrimaryFile" "argument2" "argument3" "ReferenceArray"

Arguments:

Calling listgen2 is similar to calling listgen1, but with one added argument:

1) The primary file. That is, a string of single-word references, as with listgen1;

2) A short message to be displayed to help the user. This may be left empty;

3) The button text eg: 'Ok Exit' or just 'Ok';

4) The name (only) of the reference array (this will be used to access the array).

Four arguments MUST be passed, although argument2 may be a null string.

listgen2 returns the same global variables as listgen1 and uses the same built-in functions

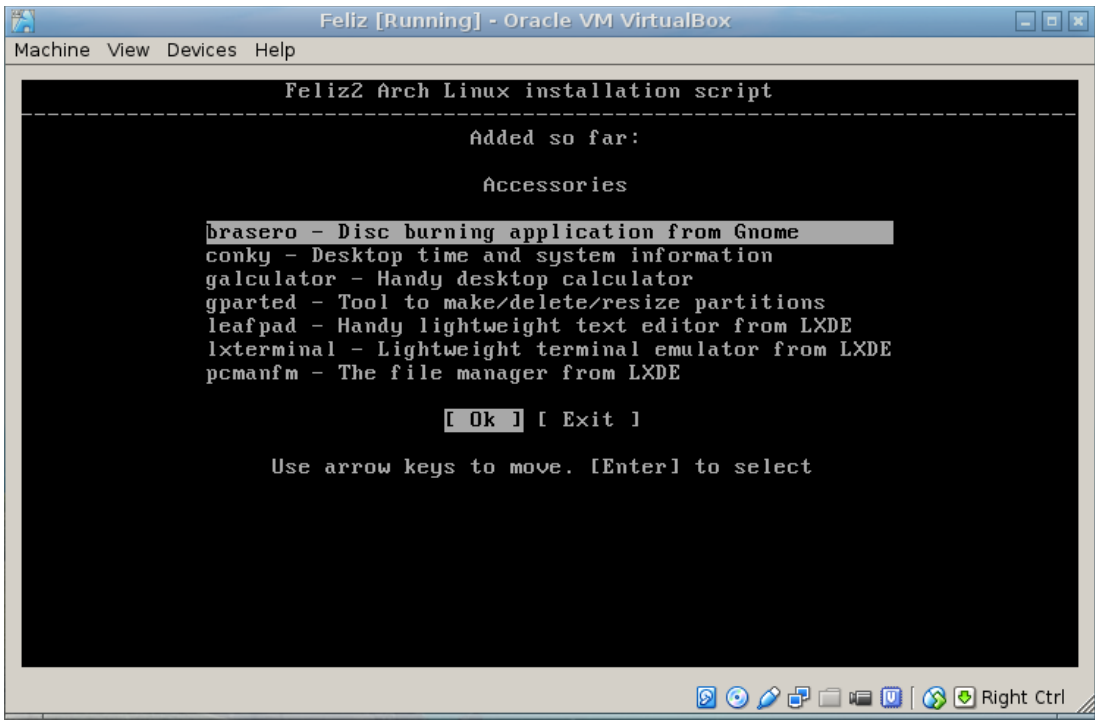
Example from Feliz

1) In this example, the variable 'Accessories' and the array 'LongAccs' are declared and initialised as follows:

```
Accessories="brasero conky galculator gparted leafpad lxterminal pcmanfm"
LongAccs[1]="Disc burning application from Gnome"
LongAccs[2]="Desktop time and system information"
LongAccs[3]="Handy desktop calculator"
LongAccs[4]="Tool to make/delete/resize partitions"
LongAccs[5]="Handy lightweight text editor from LXDE"
LongAccs[6]="Lightweight terminal emulator from LXDE"
LongAccs[7]="The file manager from LXDE"
```

They can then be used anywhere in the scripts. Here the user is offered the opportunity to choose from some of the software available from the Arch Linux repositories …

```
...
for i in ${Accessories}
do
    LongAccs1[${Counter}]="$i - ${LongAccs[${Counter}]}"
    (( Counter+=1 ))
done
listgen2 "$Accessories" "Use arrow keys to move. [Enter] to select" "Ok Exit" "LongAccs1"
```


Feliz [Running] - Oracle VM VirtualBox
Machine View Devices Help
Feliz2 Arch Linux installation script
Added so far:
Accessories
brasero - Disc burning application from Gnome
conky - Desktop time and system information
galculator - Handy desktop calculator
gparted - Tool to make/delete/resize partitions
leafpad - Handy lightweight text editor from LXDE
lxterminal - Lightweight terminal emulator from LXDE
pcmanfm - The file manager from LXDE
[Ok] [Exit]
Use arrow keys to move. [Enter] to select
Right Ctrl

listgenx

The need for a third listing function arose during development of Feliz. At one stage of the installation of an operating system it is necessary to set the location, so that the correct time is set, appropriate fonts are selected, the right languages are offered and the best mirrors are used for downloading the files needed during installation. Some of the lists generated contain more options than can be displayed in a single column by listgen1 or listgen2, and so listgenx was born.

Instead of a cursor-driven menu, listgenx generates a numbered list from which the user can choose by entering the number displayed next to the item. It will display as many columns as can be fitted onto the width of the terminal, advising the user to enter ' < ' or ' > ' to display the previous or next page of data, if relevant.

Because listgenx must redraw the screen for each pageful of data to be displayed, it must be provided with all the text it needs for user advice and prompting. As a result, listgenx requires five arguments

Before calling listgenx, the calling function must generate a file containing the items to be displayed. The items in the file must be only one word each, must be one to a line, and the file must be named input.file.

The input.file may be generated in any way that produces a list of items, one on each line:

1) It may be passed directly:

printf "%-s\n" "item1" "item2" "item3" > input.file

2) It may be generated in a loop:

```
for i in names.list
do
   echo $i >> input.file
done
```

3) It may be generated by a bash command ...

lsblk -l | grep 'part' > input.file

listgenx is called in the form:

listgenx "argument1" "argument2" "argument3" "argument4" "argument5"

argument1 is an optional string of text to be displayed at the top of the screen as information for the user. If passed as a null string, no message will be printed;

argument2 is an optional instruction line to appear at the prompt, which will be printed at the bottom of the terminal/screen, beneath the pageful of data – for example:

"Please enter the number of your selection: "

If argument2 is empty, no instruction will be printed – this could be confusing for the user, so it is advised that a message similar to that above is sent;

The remaining three arguments are paging advice:

argument3 allows for the user to back out of listgenx and return to the calling function without making a selection – for example:

"or ' ' to go back"

argument4 and argument5 inform the user that they can page forwards or back through the data by entering either '>' for the next page, or '<' for the previous page. listgenx will intelligently display these messages only if appropriate for the circumstances (ie: it will not suggest '>' if there are no more pages to be displayed. The arguments may be as follows:

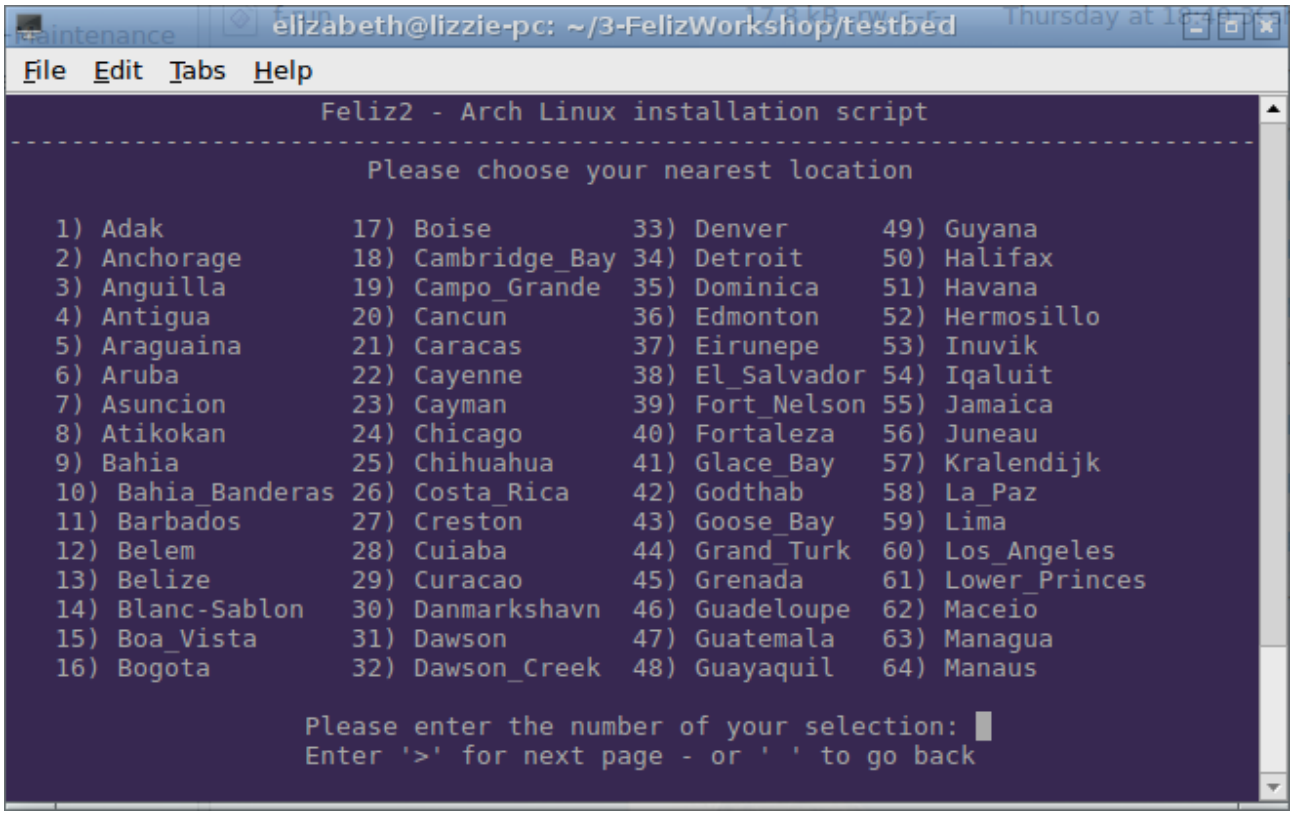
"Enter '<' for previous page"
"Enter '>' for next page"

listgenx sets the same output variables as listgen1 and listgen2

An Example from Feliz

timedatectl list-timezones | grep ${ZONE}/ | cut -d'/' -f2 > input.file # List of subzones

listgenx "Please choose your nearest location" "Please enter the number of your selection" "or ' ' to go back" "Enter '<' for previous page" "Enter '>' for next page"



Further Developments

Feliz is in constant development, and has recently become multi-lingual, but great care has been taken to keep Listgen separate from the translation code. This is so that you can use Listgen in any situation. Nevertheless, in listgen, as in Feliz, prompts and button-text were all in English, and an extensive rewrite of the listgen functions has been necessary to allow any calling functions to specify exactly the text to be displayed in and below the lists. This means that earlier editions of this guide may not reflect the changes to the calling arguments. All revisions, however, will be repeated in the 'listgen.manual' which accompanies the Feliz scripts at:

https://github.com/angeltoast/feliz2

Details of the implementation of translation capability into Feliz and Listgen will form the basis of another small book, for those of you who may like to examine and experiment with it.

Index of Some useful websites:

Awk

The AWK Manual - Table of Contents

How to use the awk language to manipulate text
An Introduction To awk

Command-line

LinuxFoundationX - Free Courses from The Linux Foundation | edX
bash Cookbook | Main / HomePage browse
LinuxCommand.org: Learn the Linux command line. Write shell scripts.
Linux Newbie Guide: Shortcuts And Commands
The 5-Minute Essential Shell Tutorial - Linux Mint Community
Terminal Command Reference
Linux Command Directory: Index
The Linux Information Project index
Introduction to UNIX and Linux: Tutorial lectures and exercise sheets
passwd - Linux Command - Unix Command
UNIX Shell Quotes - a simple tutorial
I/O Redirection
16 commands to check hardware information on Linux

Essential Commands

cut
head
tail
read
printf syntax basics
printf help and examples
tr command

General

Shell Scripting
Linux Shell Scripting Tutorial - A Beginner's handbook
Advanced Bash-Scripting Guide
LinuxCommand.org: Writing shell scripts.
Getting User Input Via Keyboard - Linux Shell Scripting Tutorial - A Beginner's handbook
Startup scripts - Linux Shell Scripting Tutorial - A Beginner's handbook
Easy Running of Scripts at Boot and Shutdown - SUSE Blog | SUSE Communities
Command Center: shell colors colorizing shell scripts Arrays
Manipulating Strings

Grep

A Beginner's Guide to Grep: Basics and Regular Expressions - Open Source For You
Linux and Unix grep Command
grep

Man-up

The man Command

Reading Man Pages

Parameter Expansion

Parameter expansion [Bash Hackers Wiki]
Bash Reference Manual: Shell Parameter Expansion

Sed

Text Manipulation with sed | Linux Journal

UNIX Command Line: Sed - save changes to same file

Unix Sed Tutorial: Append, Insert, Replace, and Count File Lines

Sed - An Introduction and Tutorial

The UNIX School: awk & sed

Frequently-Asked Questions about sed, the stream editor

sed one-liners

sed, a stream editor

Sed - help and examples

Tput

tput - Linux Command - Unix Command

Discover tput

Tput - Portable Terminal Control From Scripts