Digital Signage Project

Gert Bakker, Coleshill, UK

Table of Contents

NAME

check_read_email.sh - Start the email checking and reading process

SYNOPSIS

```
check_read_email.sh
```

DESCRIPTION

This script starts the Python program "check_read_email.py" in background if it is not yet running. If it is already active, the script exits. Therefore this script can be used to check that the Python program is still running and if not, to restart it. The Python program is executed in background so that this script can terminate.

CONFIGURATION

All configurable constants are located in the main configuration file.

OPTIONS

None

RUN OPTIONS

This script is run by CRON at regular intervals to ensure that the Python program "check_read_email.py" is still active. If not, it will be restarted. This is to ensure that the checking of emails remains active in case the program terminated.

FILES

None

ENVIRONMENT

None

SEE ALSO

None

EXIT STATUS

0 Exit OK

80 Check_read_email.py program already active

INCLUDED SOURCES

signage.conf

Main configuration constants used by this project.

CONSTANTS

LOCK_NAME

Name of the Python program to check if it is active.

FUNCTIONS

None

VARIABLES

None

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly.

First check if the Python program is active, and if so, terminate this script with exit code 80 (which is OK but can be checked).

Import the file with the configuration constants. Some of these are used by the Python program.

Start the Python program in background mode and route all STD output to be ignored.

NOTES

This script is run at regular intervals for the hours the email checking is required. That way it ensures that the email checking is kept working.

BUGS

None

EXAMPLE

This script is to be run by CRON at boot time to start the email checking and needs to run at regular intervals to ensure the program is still running. The following are examples of entries for the CRON table.

```
# Do this at boot but wait for 60 seconds before starting it to
# ensure that the system is up and running.
@reboot sleep 60 && /home/pi/Production/pi_signage/check_read_email.sh

# This entry ensures that the program (a) starts at a specific
# time and (b) checks at regular intervals that the Python
# program is still running.
0,30  7-20  *  *  *  ~/Production/pi_signage/check_read_email.sh
```

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

daily_status.sh - send a daily status email to System Administrator

SYNOPSIS

```
daily_status.sh
```

DESCRIPTION

This script does the daily cleaning and then sends the System Administrator an email with the current system status.

First it deletes any files that are over a certain number of days old.

Then it collects some health information about CPU temperature, uptime and disk utilisation percentage.

It extracts the syslog entries of interest (if the script to do this is installed). It then adds the main project log file contents.

The message file contents is then emailed to the SysAdmin.

Finally, the main project log file is cleared. A message is written to it to show that the daily status was completed. That way the log file is restarted each day.

CONFIGURATION

All configurable constants are located in the main configuration file.

OPTIONS

None

RUN OPTIONS

This script is to be run every day using CRON.

FILES

syslogquick.sh
: Extract and display the syslog messages of interest and will be added to the status email message.

pi_signage.log
: The file with today's log messages. These are added to the email message.

ENVIRONMENT

None

SEE ALSO

~/bin/syslogquick.sh
: Display Syslog entries of interest

~/bin/src/syslogcheck.conf
: Syslog configuration constants

EXIT STATUS

0 Normal Exit

INCLUDED SOURCES

signage.conf
: Main configuration of constants used by this script.

send_email.func
: Functions to send an email message either immediately or queue for later sending.

logging.func
: Message logging function.

CONSTANTS

See the configuration file for the common constants used in this script.

FUNCTIONS

deletefiles
Delete a list of files and log a message for each deleted file.

VARIABLES

MYNAME
The name of this script for use in the log messages.

EMAILMSG
Email message file name.

MY_SYSMSG
File to get the syslogquick.sh messages.

FILES2DELETE
List of all the files to be deleted when this script ends.

EMAIL_SUBJ
Email Subject line.

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Delete the aged files in the three folders that require cleaning. The number of days each file is deleted is configurable in the common configuration file. The "deletefiles" function is used to remove the files.

Start collecting the status information and store the results in variables.

Get the system start date and time using the "uptime" command and store in a variable.

Get the CPU temperature using either the "vcgencmd" command if that is available or use the "sensors" command. That way this script can run on a Linux machine or a Raspberry Pi. Useful during development.

Collect the disk usage percentage using the "df" command.

Generate the first part of the status message by outputting the above collected information.

Add the list of files in the "other message" folder. This allows the SysAdmin person to be informed that messages from an unauthorised email address were received. This is not necessarily an error, but may need to be checked.

If installed, collect the "syslogquick.sh" script output and add the syslog messages of interest to the email message. This shows what events have taken place on the device.

Finally, add the contents of the logging file to the message. When done, clear the log file and write a message to it to show that the daily status script was run.

Use the email function to send an email immediately, using the email credentials from the configuration file together with a subject line.

Delete all temporary files.

Exit the script with status code 0 (Normal exit).

NOTES

This script is to be run using a CRON entry every day.

BUGS

None

EXAMPLE

This script is run via a CRON entry. The following is an example:

```
# This entry will ensure that this script is run daily to
# provide a status message to the System Administrator.
0  1  *  *  *  ~/Production/pi_signage/daily_status.sh
```

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

dotenv - Secret data for this project

SYNOPSIS

dotenv

DESCRIPTION

The **dotenv** file is a method of storing secret information. The actual filename is, by default, ".env". The Python module "**dotenv**" allows access to this information. In this project it is used to store the email credentials and other secret information.

RUN OPTIONS

See Python '**dotenv**' module.

FILES

The name of the file used for storing the secrets.

SEE ALSO

The Python module:

dotenv

CONSTANTS

The following list all the secret items used by the Python programs and by the BASH 'logging.func' function definition.

DEBUG FLAG

This flag sets or un-sets the generation of DEBUG messages. When set to True all scripts and programs will write DEBUG log messages to the main log file. This is normally used during development, but can be useful during production to resolve problems. This flag can de set to True or False by an emailed command.

```
DEBUG='True'
```

EMAIL CREDENTIALS

These are the credentials of the email address for this system. Each system needs its own email address and this is used by the authorised senders to email commands to the system.

```
USERNAME="systemname@example.com"   Email account name
PASSWORD="the password"             Email account password
```

IMAP/SMTP SERVER DATA

These define the IMAP and SMTP server names and their PORT addresses to use in reading and sending emails. Use the email provider's IMAP server. Look for the provider's IMAP server on Google or check this page: https://www.systoolsgroup.com/imap/. For office 365, it's this:

```
IMAP_SERVER="imap-mail.outlook.com"
IMAP_PORT=993
SMTP_SERVER="smtp-mail.outlook.com"
SMTP_PORT=587
```

AUTHORISED EMAIL ADDRESSES

This is the list of email senders that are authorised to send commands to the system. Emails received from senders that are not in this list are stored in the other messages folder. There the System Administrator can then check that email if necessary. This list of authorised email senders can be changed by an emailed command.

```
EMAIL_AUTH='anyname@example.co.uk,anothername@gmail.com'
```

MESSAGE COUNTER.

This counter is used to add to the email messages file name when stored in the 'messages' folder and also to add as a prefix to the attachment files that was found in the email. After reaching the value 9990 the counter is reset. The Python program reading the emails updates this counter when it has been used for a message filename. More information is in the System Administrator documentation.

```
MESG_COUNT='1'      Message counter for use in the filename
```

NOTES

None

INSTALLATION

This file is to be installed in the production folder.

AUTHOR

Gert Bakker, Coleshill, UK

NAME

issue_command.sh - Issue a command to manage the system.

SYNOPSIS

`issue_command.sh` *`command`* `[`*`command`* `...]`

DESCRIPTION

This script issues commands to manage the presentations and the Energenie power sockets. The following commands are currently implemented:

stop Stop the current presentation

restart Restart the last presentation

start Same as restart

reboot Reboot the device

shutdown
: Shutdown the device

run Run a presentation saved in the "permanent" folder

turn_on N
: Turn the given Energenie power socket ON. Socket number needs to be provided [0-4].

turn_off N
: Same as turn_on, but turning the power socket OFF

help Display a help screen and exit the script.

When the above commands are processed the script to start a presentation is run in background mode.

CONFIGURATION

All configurable constants are located in the main configuration file.

OPTIONS

command
: Command to control the presentation and the device. See the list in the HELP screen for the possible commands.

RUN OPTIONS

This script is intended to be run from a *command* prompt, in other scripts and/or in CRON entries at scheduled times.

FILES

turn_on_off.py
: Python program to turn Energenie power sockets ON and OFF.

SEE ALSO

turn_on_off.py

EXIT STATUS

0 Normal exit

1 Error detected. See the logging file for the cause.

INCLUDED SOURCES

signage.conf
: Main configuration of constants used by this script.

logging.func
: Message logging function.

CONSTANTS

See the configuration file for the constants used in this script.

FUNCTIONS

err() Display an error message to the console (STDOUT). Useful when running this script manually from the system prompt. Only used when the argument is not recognised.

VARIABLES

MYNAME

The name of this script for use in the log messages.

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the configuration and function files.

Check that there are arguments. If none found, display an error message and exit the script with status code 1.

The script can execute multiple commands. Make sure they are in a logical sequence and that the additional data is provided.

The script now checks the argument and processes them as follows:

stop Write the *command* STOP to the running next trigger file and write a log message that the *command* "stop" was performed.

restart or start

Nothing needs to be done as the script to start the presentation will automatically run whatever is in the running now trigger file. Write a log message that a restart or start was requested.

reboot Write a log message and reboot the system.

shutdown

Close the presentation, the email checking process and turn off all Energenie power sockets. Sync the files one last time, wait 10 seconds to make sure everything has finished and shutdown the device.

run This *command* requires another argument which is the name of the file to run. This argument must be a file in the permanent folder, but without the folder name, just the filename. If the filename is in the permanent folder, the file is copied to the running next trigger file. Messages are logged if this is successful or if the file was not found.

turn_on

Turn ON an Energenie power socket given as a parameter [0-4].

turn_off

As for turn_on, this will turn off the Energenie power socket as given in the next parameter [0-4].

help Display a help screen with the possible commands in them. Exit the script with code 0.

Unknown *command*

If the *command* argument does not match the above possibilities, display an error message to the console and write an error message to the logging file.

After the commands have been processed, the script that runs the presentation is started in background mode.

Exit the program with code 0

NOTES

None

BUGS

None

EXAMPLE

This *command* can be run from a *command* prompt as follows:

```
issue_command.sh start turn_on 0
```

The following is an example of a CRON entry to stop the current presentation at a scheduled time.

```
0  21  *  *  *  ~/Production/pi_signage/issue_command.sh stop
```

INSTALLATION

This script needs to be installed in the production folder and in a folder that is in the PATH variable. It is recommended to install it in the '~/bin' folder or the /usr/local/bin folder. That way it can be executed as a shell *command*.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

kill_check_email.sh - Kill the check_read_email program

SYNOPSIS

```
kill_check_email.sh
```

DESCRIPTION

This script terminates the Python program "check_read_email.py" so that the program terminates properly.

CONFIGURATION

All configurable constants are located in the main configuration file.

OPTIONS

None

RUN OPTIONS

Run this script at the end of the day when the Python program needs to stop working.

FILES

None

ENVIRONMENT

None

SEE ALSO

pkill Utility to send a signal to a running process.

EXIT STATUS

0 Normal exit

INCLUDED SOURCES

signage.conf

Main configuration constants used by this project.

CONSTANTS

PROGRAM_NAME

Name of the Python program to send a signal to.

FUNCTIONS

None

VARIABLES

None

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly.

Include the file with the constants for this project.

Using the "pkill" utility, send the SIGNAL 2 (INT) to the Python program so that it terminates properly. Ignore any errors.

NOTES

Run this script at the time of the day when the Python program needs to be terminated.

BUGS

None

EXAMPLE

The following is an example of the CRON entry to terminate the email checking program that is running in background.

```
10  21  *  *  *  ~/Production/pi_signage/kill_check_email.sh
```

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

logging.func - Shell function to write a message to the log file

SYNOPSIS

```
logging.func LOGLEVEL log_message_text ...
```

DESCRIPTION

This mimics the logging feature of the Python logging module. It therefore does not log messages with a certain log level. If the *LOGLEVEL* is "DEBUG" and if the "dotenv" DEBUG flag is not True, do not write the message to the log file. All other messages are written to the log file using the same format as is used in all the Python programs. The message format is similar to the SYSLOG messages.

CONFIGURATION

None

OPTIONS

LOGLEVEL

Logging message level such as DEBUG, ERROR, INFO, etc.

log_message_text

Text of the message to write to the log file.

RUN OPTIONS

Include this function into all scripts that need to write log messages

FILES

dotenv The dotenv secrets file. Used for the DEBUG flag.

ENVIRONMENT

None

SEE ALSO

dotenv Secrets configuration file to get the DEBUG status value.

signage.conf

This must be sourced in by the scriptfor this function to work.

EXIT STATUS

None

INCLUDED SOURCES

dotenv - Sourced in to get the DEBUG flag value.

CONSTANTS

See the configuration file for the constants used in this script.

FUNCTIONS

See MAIN PROCESSING

VARIABLES

None

MAIN PROCESSING

From the dotenv configuration settings, get the DEBUG value. If the DEBUG value is not True and the *LOGLEVEL* is "DEBUG", exit the function as this level is not logged if this case. Using the *LOGLEVEL* and *log_message_text*, write a message to the log file using the same format as is defined for the Python programs.

Exit the function.

NOTES

None

BUGS

None

EXAMPLE
See RUN OPTIONS

INSTALLATION
This script is installed in the main production folder.

FUTURE
None

AUTHOR
Gert Bakker, Coleshill, UK

NAME

process_messages.sh - Process the retrieved email messages

SYNOPSIS

```
process_messages.sh
```

DESCRIPTION

This script starts the Python program that processes the email messages that have been read and saved by the check_read_email program. The contents of the messages is read and this will create trigger files to do SysAdmin tasks and/or to start/stop a presentation. This script will run the scripts to perform the SysAdmin tasks and/or to run a presentation.

CONFIGURATION

See the configuration file with the constant definitions.

OPTIONS

None

RUN OPTIONS

This script is run by the Python program check_read_email.py only.

FILES

see INCLUDED SOURCES

ENVIRONMENT

None

SEE ALSO

signage.conf

See the documentation for the configurable elements for this project.

EXIT STATUS

0 Normal exit

INCLUDED SOURCES

signage.conf

Constant values for use in all script for this project.

logging.func

Function to log messages to the log file.

CONSTANTS

DEBUG

Set to logging a DEBUG message.

MYNAME

Set to the name of the script. Use in logging messages.

FUNCTIONS

None.

VARIABLES

None

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the source files to get the configuration constants and the DEBUG status setting.

Run the Python program to process the saved email messages and attachments. This program generates trigger files to start the system admin task and/or the presentation.

If the "manage next" trigger file is present, execute the script to run the system admin tasks and wait until the tasks have been completed. This way the system admin is done before running a presentation.

If the "running next" trigger file is present, execute the script to start the presentation.

Terminate the script with exit code 0.

NOTES

None

BUGS

None

EXAMPLE

This script is run by the check_read_email.py program only.

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

send_email.func - Function definitions for emailing a message

SYNOPSIS

```
source send_email.func
```

DESCRIPTION

This script defines two functions for sending an email message.

Both functions generate an RFC 5322 compliant email message from the arguments passed. It stores the message in the queue folder for sending.

The first function generates the email message, moves it to the queue folder and then exits the function. The sending of the message will be done by the regular send script.

The second function does the same as the first function, but once the message has been queued, it executes the script to send the email message. This does mean that any queued messages are also send.

Note that these functions depend on the configuration settings in the signage.conf file. See the RUN OPTIONS.

OPTIONS

The two functions are used as follows:

```
queue_email "To_adrs" "Subject" "Message_File_name" ["attachfile]"

send_email_now "To_adrs" "Subject" "Message_File_name" ["attachfile]"
```

Where:

queue_email and send_email_now
: The names of the functions.

TO_adrs
: The address to send the email to.

Subject
: The text of the subject for the message.

Message_File_name
: The file that has the message body in.

attachfile
: Filename of the attachment to the message (optional).

RUN OPTIONS

Any script that sources in these functions must also have sourced in the main configuration file. Data in the latter is used by the former.

SEE ALSO

send_email.sh
: Script to email an RFC5322 formatted message.

signage.conf
: The main configuration constants.

FUNCTIONS

This script defines two functions to queue an email message for later sending or to send it now.

VARIABLES

QUEFILE
: The temporary filename for the message to be queued.

MAIN PROCESSING

The following describes what each function does. They each format the RFC5322 email message in a temporary file. Each function will move the message to the queue folder and, if sending it now, will run the

script to send the email message.

queue_email()

This function queues the RFC5322 message for later sending. Create a unique file name for the message to be emailed. Generate the message header lines using the "To_adrs" and "Subject" arguments passed to this function and write them to the temporary message file. If the attachfile argument was given, create the message header for this. Append the contents of the message file from the "Message_File_name" argument to the temporary message file. Move the temporary message file to the email queue folder. There the message awaits delivery when the send email script is next executed.

send_email_now()

This function queues the RFC5322 message for immediate sending. This function does the same as the function to queue a message. Once the message has been queued, run the script to send the messages in the queue in background mode.

NOTES

This script assumes that when it is used in a script that the main configuration file has also been sourced in.

The functions in this script also rely on the scheduled sending of queued emails. This is normally done via a CRON entry and is implementation dependent.

EXAMPLE

send_email_now "to@example.com" "Subject text" "message" ["attachfile"]

INSTALLATION

This script is installed in the production folder

AUTHOR

Gert Bakker, Coleshill, UK

NAME

send_email.sh - Send emails in RFC 5322 formatted files

SYNOPSIS

```
send_email.sh [message_file]...
```

DESCRIPTION

This script sends one or more email messages that are RFC 5322 formatted.

If arguments are present, these email messages files are emailed one by one and each message file moved to the unqueue folder after successful sending. If the message failed to be send, the message file will be moved (or kept in) the queue folder, where it will be retried when this script runs again.

If no arguments present, the queue folder is checked and any files in this folder are emailed. Again, if successful, the message file is move to the unqueue folder. (See main documentation for the use of this folder).

The sending of the messages is done by the Python program "send_email.py".

This script can therefore be used to email a message immediately or to send messages from the queue folder at regular intervals.

CONFIGURATION

See the configuration file.

OPTIONS

message_file

The name of the RFC5322 formatted message file. TO, SUBJECT and ATTACHMENT are taken from the header lines in this file. The FROM is generated by the Python program from the main configuration constants.

RUN OPTIONS

This script is either run as and when an email needs to be send or by a regular CRON entry (say every 5/6 minutes). The script works with or without arguments.

FILES

send_email.py

Python program to email the message files passed as arguments to this program.

ENVIRONMENT

None

SEE ALSO

send_email.py

The Python program that does the actual emailing of the RFC 5322 formatted messages.

EXIT STATUS

0 Normal exit

INCLUDED SOURCES

signage.conf

Main configuration file with constants used by this script and any Python programs.

logging.func

Function to write messages to the logging file.

CONSTANTS

DEBUG

Set this to logging DEBUG messages.

MYNAME

The name of this script. Used in the logging messages.

FUNCTIONS

None

VARIABLES

FILES List of message files found.

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the files so that the configurable constants and the required functions are included.

If there are any arguments, collect all the arguments into an array called "FILES".

If there are no arguments, find all the files in the queue folder and store the filename in the array called "FILES". If no files were found in the queue folder, exit the program with status 0. This is normal and means that this script can run using CRON entries.

The variable FILES now contains names of message files that can be emailed.

Execute the Python program "send_email.py" together with the list of filenames as arguments to do the actual emailing. If the outcome from the program is not 0 (zero), log an error message. This alerts the SysAdmin that there were problems with that instance.

Exit the script with status code 0 (normal exit).

NOTES

This script is designed to be run by other scripts and/or by a CRON entry for regular sending of queued messages.

BUGS

None

EXAMPLE

Cron entry example for sending email messages every 6 minutes. # Send emails from the queue folder 1-59/6 7-18 * * * ~/Production/pi_signage/send_email.sh

INSTALLATION

This script is installed in the production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

signage.conf - Main configuration constants for this project.

SYNOPSIS

```
signage.conf
```

DESCRIPTION

This is the sourced-in file with the main configuration constants. This file is used by all the scripts and Python programs of this project. It defines the main constants for this project. If a Python program cannot find a configuration constant, it will use a preset default value. (See the Python programs for these).

Before putting this project into production, adjust all the constants to suit the specific implementation of this project. This may include changing the defaults in the Python programs!

All constants must be defined with the "export" command to ensure that the Python programs can use them as well. For example:

```
export PI_SIGN_LOG='pi_signage.log'
```

Of course, do not change the name of the constants.

RUN OPTIONS

source (.) signage.conf

CONSTANTS

The constants listed below are to be adjusted for the specific implementation of this project. Ensure that each constant is preceded with the "export" command.

LOGGING FILE NAME

The name of the file that will store the log messages. Define the filename to use for logging messages that are generated by the scripts and the Python programs. This file will get all the ERROR and INFO messages generated by this project. During development it will also contain the DEBUG messages. The file is emailed daily to the System Administrator and then emptied for the next day's messages.

```
PI_SIGN_LOG='pi_signage.log'
```

RUNNING FILES

These are the files that trigger specific actions. These files are used by scripts to start a presentation and/or system admin tasks. See the System Documentation for a full explanation.

```
RUN_NEXT='running_next'          Next presentation to run
RUN_NOW='running_now'            Presentation that is running now
RUN_LAST='running_last'          Last presentation
RUN_ADM='manage_next'            System Admin task command
```

FOLDER NAMES

These are the folders used by this project. See the System Documentation for more information on the use of these folders.

```
ATCH_FOLDER="./attachments"      Attachments are stored here
MESG_FOLDER="./messages"         Email messages are stored here
QUEUE_DIR="./queue"              Messages to email are stored here
UNQUEUE_DIR="./unqueue"          Messages that have been emailed are here
OTHR_FOLDER="./other_msg"        Messages from unauthorised senders
PERM_FOLDER="./permanent"        Permanent presentations are stored here
```

EMAIL ADDRESS

Email address of the System Administrator. This email address receives the daily status email from the system and other emails that were generated by the system tasks.

```
SA_EMAIL_TO='My name <my-name@example.com>'
```

EMAIL SCRIPT NAME

The name of the script that will send the emails. Used by scripts that are sending email messages.

```
EMAIL_SCRIPT="./send_email.sh"
```

FILE DELETION PERIODS

The number of days to wait before deleting files. Set the number of days after which the files in these folders are to be deleted.

```
FDEL_ATCH=+20                   Attachment folder files
FDEL_UNQU=+7                    Unqueue folder files
FDEL_OTHR=+7                    Other message folder files
```

NOTES

None

INSTALLATION

This file is installed in the production folder.

AUTHOR

Gert Bakker, Coleshill, UK

NAME

start_presentation.sh - Start or stop a presentation.

SYNOPSIS

```
start_presentation.sh [wait] [ON] [OFF]
```

DESCRIPTION

This script is the only one that starts or stops a presentation.

Presentations depend on the contents of trigger files. See the main system documentation on the principle behind these trigger files. This script is to be run when the LXDE GUI starts after a boot-up and/or by any script that wants to start or stop a presentation. The *wait* argument provides a *wait* time after boot-up so that the network can become stable. It can also turn on and off all the Energenie power sockets that control the power to the screens, but only if these are used by the implementation.

OPTIONS

wait Permits a sleep time of 10 seconds. To be used when the script is executed by the LXDE GUI interface at start-up. If need be, this option can be repeated to get multiple 10 second sleep timeouts.

ON Turn on all the Energenie power sockets.

OFF Turn off all the Energenie power sockets.

RUN OPTIONS

This script is intended to be run by CRON, LXDE GUI start-up and other scripts in this project. This script is NOT intended to be run manually from the command line.

FILES

signage.conf

Main configuration of constants used by all scripts and Python programs.

logging.func

Defines a shell script based logging function (emulating the Python module "logging") to be used to write log messages to the log file for this project.

SEE ALSO

None

EXIT STATUS

0 Normal exit

1 Running now trigger file was not found

INCLUDED SOURCES

signage.conf

Main configuration of constants used by this script.

logging.func

Message logging function.

CONSTANTS

DISPLAY

Set the display environment variable used to blank or turn on a screen.

MYNAME

Set to the name of this script for use in the log messages.

BROWSER

Set to the name of the Chromium browser for this device.

FUNCTIONS

None

VARIABLES

None

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the configuration constants and the logging function.

If there are any arguments passed with this script, process these as follows. If the option "*wait*" is found, pause the execution of this script for 10 seconds. Note that this option can be used more than once if more delay is required. If the option is "on" or "off", send a command to all the Energenie power sockets to turn them *ON* or *OFF* respectively.

Find out what the name of the Chromium browser is on the device this script is running. This script was developed on Linux Mint but runs in production on a Raspberry Pi. The name of the browser is different in these environments.

Test if the 'running next' trigger file is present. If so, move the 'running now' file to the 'running last' file, therefore preserving the last presentation. Move the 'running next' file to the 'running now' trigger file. The 'running next' trigger file has now disappeared, which is the intention.

Get the 'running now' information by sourcing-in the file. If the 'running now' file is not found, log an error message and terminate the script with exit code 1.

Kill the current presentation. Forcefully kill the Chromium and PowerPoint processes. This is for the current implementation of this project. More kill commands may need to be added depending on the implementation.

Now the script has the command to start a presentation.

If the command value is "Google", first reset the status messages in the chromium preferences file to avoid error messages being displayed when chromium starts. Then execute the browser command with the options to make it work in kiosk mode. Use the Google URL passed in the DATA part of the trigger file. Start the browser in background mode.

If the command value is "PowerPoint" or "Impress", find where the attachment file is by checking both the 'attachments' and 'permanent' folders for the filename given in the DATA variable. In order for the attachment file to not be accidentally deleted, the modify time for the file is set to now (ie. touched) if it is in the attachments folder. Wherever the file is, execute the LibreOffice program with the attachment filename as the argument and start this in background mode.

If the command value is for "Stop" or "Halt", it means the presentation must be stopped. This has already been done above. Copy the 'running last' trigger file to the 'running now' file so that when this script is run again, the last presentation will be restarted. This makes it easy to start and stop a presentation and not losing it's contents.

If the command value is not one of the above, log an error message.

NOTES

This script is not intended to be used manually from a command prompt. It is to be used by CRON, other scripts and/or the LXDE start-up process.

BUGS

None

EXAMPLE

See the RUN OPTIONS

INSTALLATION

This script is installed in the production folder.

FUTURE

There may be other presentation functions in the future and these will require this script to be modified to accommodate them.

AUTHOR

Gert Bakker, Coleshill, UK

NAME

sysadmin_download.sh - Email any file to the SysAdmin

SYNOPSIS

`sysadmin_download.sh` *`file_name`*

DESCRIPTION

This script is run by the System Admin tasks in order to attach any file to an email. This is so that the System Admin can get a copy of the installed file in case any amendments need to be made to it. See the System Administration specification in the System Manual.

CONFIGURATION

None

OPTIONS

file_name

Name of the file to attach to the email

RUN OPTIONS

This script is started by the SysAdmin task script when needed. It is not intended to be run as a command.

FILES

None

ENVIRONMENT

None

SEE ALSO

None

EXIT STATUS

0 Normal exit

INCLUDED SOURCES

signage.conf

Main system configuration constants.

logging.func

Function to write logging messages to the main log file.

send_email.func

Functions to send an email message either immediately or queue for later sending.

CONSTANTS

MYNAME

The name of this script. Used in the logging messages.

MY_DEBUG

Command to use for the DEBUG messages.

EMAIL_DATA

Email message body text.

EMAIL_SUBJ

Email message subject line.

FUNCTIONS

None

VARIABLES

None

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

If the argument is for "crontab", get the content of the CRONTAB entries into the file named "crontab.txt". Send the email with the crontab file as an attachment using the email function.

If the argument is for any other file, use the email sending function to send the email with the named file as an attachment.

There is no checking that the argument is a valid file name as that has already been done by the process that started this script.

NOTES

File name checking takes place in the process that started this script. Therefore there is no file checking done in this script.

BUGS

None

EXAMPLE

See RUN OPTIONS

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

sysadmin_list.sh - Perform the LIST SysAdmin task

SYNOPSIS

```
sysadmin_list.sh list_of_folders ...
```

DESCRIPTION

This script processes the "list" command and is started by the SysAdmin tasks script.

The additional data supplied with this command are the folders for which the listing is requested. These listing are then emailed to the system administrator email address in the configuration file.

CONFIGURATION

See the configuration file.

OPTIONS

list_of_folders
: List of the folder names to email the contents of.

RUN OPTIONS

This script is executed by the System Admin main task script and is not intended to be run manually.

FILES

None

ENVIRONMENT

None

SEE ALSO

None

EXIT STATUS

0 Normal exit

1 Argument error

INCLUDED SOURCES

signage.conf
: Main configuration of constants used by this script.

logging.func
: Message logging function.

send_email.func
: Functions to send an email message either immediately or queue for later sending.

CONSTANTS

MYNAME
: Set to the name of this script for use in logging messages.

MY_DEBUG
: Set to the command to log a debug message.

EMAIL_DATA
: File to store the email message in.

EMAIL_SUBJ
: Subject line for the email message.

FUNCTIONS

None

VARIABLES

array Array with the requested folder names.

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the files so that the configurable constants and the required functions are included.

Check that there is at least one argument passed with this script. If none found, write an error message to the logging file and exit the script with status code 1.

Using the read command, get the arguments into an array for processing.

Going though all the folder names in the array, if the folder name is an existing folder, get the full listing of the files and write to the email data file. For creating a nice format, make sure that the second folder list has a blank line in between the two listings. If the argument is not an existing folder, write an error message to the log file.

When all elements in the array have been processed, send the email immediately using the "send email now" function found in the sourced-in file.

NOTES

None

BUGS

None

EXAMPLE

See RUN OPTIONS

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

sysadmin_tasks.sh - Perform the SysAdmin tasks for this project.

SYNOPSIS

```
sysadmin_tasks.sh
```

DESCRIPTION

This script processes the System Administration commands that have been emailed to the system. It is the main script that deals with the SysAdmin tasks for this project. It executes other scripts and Python programs to perform some of the admin tasks as required.

See the main documentation for more information about which commands are possible to administer the system,

CONFIGURATION

See the configuration file.

OPTIONS

None

RUN OPTIONS

This script is started by the script that processes the emails.

FILES

dos2unix
: DOS to Unix and vice versa text file format converter.

ENVIRONMENT

None

SEE ALSO

None

EXIT STATUS

0 Normal exit

INCLUDED SOURCES

signage.conf
: Main configuration of constants used by this script.

logging.func
: Message logging function.

send_email.func
: Functions to send an email message either immediately or queue for later sending.

CONSTANTS

MYNAME
: Set to the name of this script for use in logging messages.

MY_DEBUG
: Set to writing DEBUG messages to the log file but only if the DEBUG flag is "True".

EMAIL_DATA
: Filename for the message to be emailed at the end of this script.

FUNCTIONS

None

VARIABLES

EMAIL_SUBJ
: Subject line for the message to be emailed.

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the files so that the configurable constants and the required functions are included.

Check that there is a management trigger file. If not there, exit the script with status code 0. Write a log INFO message as this is actually not supposed to happen. This script is called only when this trigger file is present.

Source in the trigger file that contains three variables.

The emailed command is now interpreted as follows:

List This means that a list of one or more folders has been requested. Start the script that performs this task.

update This means that a new file is available to replace an existing file. This can be a script, Python program or other data file. Start the script that processes this request.

reboot This means that a reboot has been requested. Kill the email check program and issue a command to stop the presentation. If the python program "turn_on_off" is present, turn off all the Engergenie power sockets. Sync the files and sleep for 10 seconds to stabilise things. Log the event and then execute the reboot (or shutdown) command.

shutdown
This means a shutdown of the system was requested. Perform the same tasks as for "reboot" but do a shutdown instead.

screen This means that a request to turn one or all Energenie power sockets on or off. The additional data indicates if the power goes on or off. Only do this if the python program 'turn_on_off' is present and executable.

save_now
This means that a request is received to save the running now presentation. The additional data is used as the filename of the saved information. Execute the Python program to save the running_now file to the permanent folder.

crontab
This means that the attachment is a replacement of the crontab for this user. The attachment is installed by the crontab utility after ensuring the attachment is in LINUX text format.

download
Email a file as an attachment to the SysAdmin. This task is done by a separate script.

delete Delete a file in the "permanent" folder only. There can be more than one file to be deleted. Delete each file and log the delete and add the delete to an email message.

When the trigger file has been processed, delete the trigger file.

If the email message file has any text in it, send the email to the SysAdmin using the subject line created by this script.

Exit the script with status code 0 (normal exit).

NOTES

None

BUGS

None

EXAMPLE

See RUN OPTIONS

INSTALLATION

This script is installed in the main production folder

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK

NAME

sysadmin_update.sh - Perform the UPDATE SysAdmin command

SYNOPSIS

```
sysadmin_update.sh data_string attachment_name
```

DESCRIPTION

This script processes the "update" command, which means it is going to replace an existing file with a new one that was emailed as an attachment.

This way scripts, programs and configuration files can be updated by simply sending an email with the new file.

Only exiting files can be replaced. Any new files will need access to the system and requires knowledge of the project and installation.

CONFIGURATION

See the configuration file.

OPTIONS

data_string
: The name of the file to be replaced.

attachment_name
: The name of the attachment that is to replace the file named in the *data_string*.

RUN OPTIONS

This script is executed by the System Admin main task script and is not intended to be run manually.

FILES

dos2unix
: Convert DOS to UNIX line ending

ENVIRONMENT

None

SEE ALSO

None

EXIT STATUS

0 Normal exit

1 Invalid number of arguments

INCLUDED SOURCES

signage.conf
: Main configuration of constants used by this script.

logging.func
: Message logging function.

send_eail.func
: Function to send an email message.

CONSTANTS

MYNAME
: Set to the name of this script for use in logging messages.

DEBUG
: Set to the command to log a debug message.

EMAIL_DATA
: File name for receiving the status message in.

EMAIL_SUBJ
Subject line for the message to email.

OLD_DIR
Working folder name used for testing.

NEW_DIR
Production folder name to use in all scripts. Note that the layout for these two constants must be maintained as is. Only change the value if need be.

FUNCTIONS

None

VARIABLES

CHMOD
File permission of the file to be replaced.

MAIN PROCESSING

Set the working folder to the proper folder to ensure all following statements are executed properly. Set the MYNAME variable to the script name.

Source in the files so that the configurable constants and the required functions are included.

If there are no 2 arguments, write an error message to the log file and exist the script with status code 1.

The argument *data_string* must be an existing file in the main production folder. If this is the case, check that the *attachment_name* argument is an existing attachment file. If the attachment exists, get the file permissions of the original file and save in a variable. Move the attachment file to the file to be replaced. Using the "dos2unix" utility, correct the new file with the LINUX line endings. Correct the working folder in the new file to the production folder to ensure that the script/program uses the production folder. Correct the file permissions with what the original file had (see above). Log the replacements and add it to the message to be emailed.

In case *data_string* is not an existing file or *attachment_name* is not an attachment file, write an error message to the log file and to the email message.

If the email message file contains any text, send the email message to the system administrator immediately.

Exit the script with status code 0.

NOTES

The exit status code of this script is not used by any script that executes this script. It is added for future use.

BUGS

None

EXAMPLE

See the RUN OPTIONS

INSTALLATION

This script is installed in the main production folder.

FUTURE

None

AUTHOR

Gert Bakker, Coleshill, UK