Introduction

**Last updated 04 May 2016**

Yet Another Dialog - YAD is a simple tool for developing Graphical User Interfaces.

It's similar to Zenity but with lot more functionality.

It's a single binary and only about 144k for the Fatdog64 version, about 150k for tahrpup.

It will integrate with bash and gtkdialog scripts.

In it's simplest form, type **yad** in a terminal to get:


This help has been developed in the same order as yad's internal help.

Type **yad --help-all** to see all command line help.

Unfortunately this help does not provide enough information to use yad to it's full potential. This is why I developed this tutorial.

I hope you find it usefull.

Code is highlighted in **blue,** warnings in **red** and Tips in **magenta.**

Comments and examples can be sent to **smokey01@internode.on.net**

Cheers

Smokey01 (Grant)

If you type **yad --help-all > /root/yad-help.txt** into a terminal it will make a text file containing the built-in help.

**Tip: I like to use Notecase to develop Help manuals because it's easy to use, has a nice interface and keeps all the data in one file. The file is actually HTML compliant which means you can view the file in your browser. Just make a copy of yad.ncd and call it yad.html.**
**All of the images in the file are saved as base64 format. If you wish to make a copy of any of the images, right mouse click and select Save Picture.**

I generally keep a copy of this file at dttp://smokey01.com/yad

It may not be formatted all that well in the browser as I've desingned it for Notecase but the information can easily be found with CTRL-F.

Credits

Thanks to the following contributors participating on the Puppy Linux Forum: in order of appearance.
smokey01
CatDude
rg66
Geoffrey
L18L
kjdixo
slavvo67
mikeb
stemsee
seaside
Bert
rcrsn51
step
Argolance

**Victor Ananjevsky (YAD Creator)**

Resources

Get the latest copy of this YAD help here.

Where to obtain a copy of YAD?
https://sourceforge.net/projects/yad-dialog/

Once you have downloaded the source code it needs to be compiled. You need to have the development tools loaded. In Puppy Linux, simply load the Devx.

Once you have downloaded the source code which should look something like yad-0.36.0.tar.xz. Extract it and open a terminal in the extracted directory.

type the following commands, one at a time:

**./configure --prefix=/usr**
**make**
**make install**

You don't actually need to type make install. Simply copy the yad binary to one of your bins, EG: /usr/bin

You can make the yad binary smaller by stripping it.
**strip --strip-unneeded yad**

That's it, you're ready to use Yad.

Another good place to get some help about Yad is:
https://groups.google.com/forum/#!forum/yad-common

Also on the Puppy lInux forums at:
http://murga-linux.com/puppy/viewtopic.php?t=97458

General Options

This is where you will find the general options which contain the majority of commands
that you would likely use on a daily basis.

Each of the options are explained below in the sub menus.

In each case I will try to provide an example and some form of output.

This will allow you to cut & paste the code to save you some time and
also see the result of the command.

--title

A custom title can be added to the bar by adding the **--title** option.

**yad --title "This is my title"**


You will notice all the text in the default dialog does not fit.

To make the dialog wider, use the **--width** option.

--window-icon=ICONPATH

This allows you to choose your own icon.

You may choose one from the available icons in gtk-dialog such as:

**yad --window-icon=gtk-yes**


Which will give you a green ball on the title bar.

You can also select your own icon by defining the path to it.

**yad --window-icon=/root/up.png**


Of course for this to work the icon file must be available.

--width=WIDTH

To make the dialog a bit wider add the **--width** option

**yad --title "This is my title" --width=200**


--height=HEIGHT

There is also a height option.

**yad --title "This is my title" --width=200 --height=200**


--geometry=WxH+X+Y

This command allows you to set the Width and Height and where on the screen to place the dialog.

For example: Suppose you want to create a dialog 400 pixel wide, 400 pixels high dialog at location X in pixels 512 and Y at 384 pixels.

If your screen resolution was 1024 x 768 then the top left corner of your 400 x 400 dialog should be in the centre of your screen.

**yad --geometry=400x400+512+384**

Now if you wanted the centre of your dialog to be in the centre of your screen you would type:

**yad --geometry=400x400**

Simply leave off the X and Y coordinates. X=horizontal, Y=vertical.

Just remember, there are minimum defaults. You can't make the dialog smaller than the default size.

Try **yad --geometry 1x1**


As you can see it's clearly bigger than 1x1.

**Be careful though, because it will let you define a much bigger dialog than your screen size.**

**If you try yad --geometry 3000x3000 your complete screen will be consumed unless your screen size is bigger than 3000x3000.**

Press the Enter Key to return to normality.

--timeout

The **--timeout** command closes the dialog after a specified time.

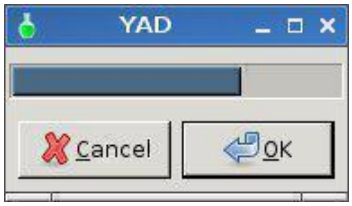
**yad --timeout=10** will automatically close the dialog after 10 seconds.

You can press Enter, click on Ok or Cancel to close it quicker.


--timeout-indicator=POS

This will display a dialog and also a countdown progress bar, very cool.

**yad --timeout=10 --timeout-indicator=top**




You need to use the --timeout command with the --timeout-indicator

You will find many commands depend on others as you work through the tutorial.

--text=TEXT

Place text on the dialog.

**yad --text="Good Morning"**


--text-align

To justify text use **--text-align=TYPE** where type can be left, right or center.

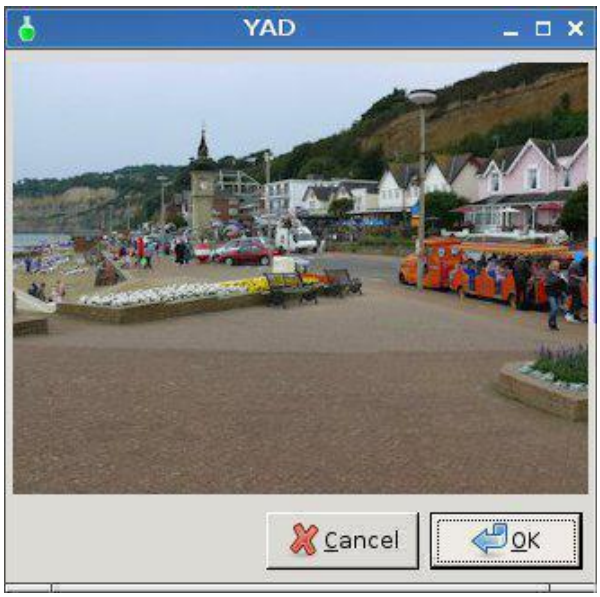
**yad --text="Good Morning" --text-align=center**


You need to use the **--text-align** command with **--text** command.

--image=IMAGE

This will allow you to place an image in your dialog box.

**yad --image /root/iow2.jpg**




--image-on-top

**--image-on-top** Show image above the main widget

**yad --image=info --text="widget is right of image" --entry**


**yad --image=info --image-on-top --text="widget is bottom of image" --entry**


It has nothing to do with the dialog having focus like **--on-top**

--icon-theme=THEME

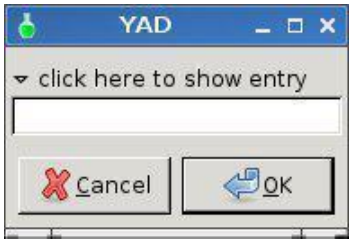
**--icon-theme=THEME** Use specified icon theme instead of default

Mmm still struggling with this
--expander=TEXT

**--expander=TEXT** shows hidden text.

**yad --entry --expander="click here to show entry"**


Click on the twistie to show hidden entry box.



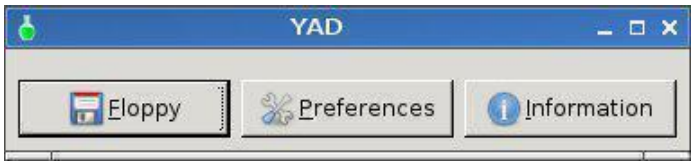

Whatever you type is parsed like the script below.

```
#!/bin/sh
ANSWER=`yad --entry --expander="click here to show entry"`
yad --text="You typed $ANSWER in the entry box"
--button=NAME:ID
```

This command allows you to add your own buttons to the dialog.

Let's create three button. We will call them Floppy, Preferences and Information.

**yad --button=gtk-floppy:0 --button=gtk-preferences:0 --button=gtk-info:0**




It retrieves the default gtkdialog icons.

To see what is available paste the below into a script.

```
#! /bin/bash
export MAIN_DIALOG='
<vbox>
<frame Stock>
<tree rules_hint="true" exported_column="1">
<height>400</height><width>250</width>
<label>Stock ID</label>
<item stock="gtk-dialog-authentication">gtk-dialog-authentication</item>
<item stock="gtk-dialog-info">gtk-dialog-info</item>
<item stock="gtk-dialog-warning">gtk-dialog-warning</item>
<item stock="gtk-dialog-error">gtk-dialog-error</item>
<item stock="gtk-dialog-question">gtk-dialog-question</item>
<item stock="gtk-dnd">gtk-dnd</item>
<item stock="gtk-dnd-multiple">gtk-dnd-multiple</item>
<item stock="gtk-about">gtk-about</item>
<item stock="gtk-add">gtk-add</item>
<item stock="gtk-apply">gtk-apply</item>
<item stock="gtk-bold">gtk-bold</item>
<item stock="gtk-cancel">gtk-cancel</item>
<item stock="gtk-cdrom">gtk-cdrom</item>
<item stock="gtk-clear">gtk-clear</item>
<item stock="gtk-close">gtk-close</item>
<item stock="gtk-color-picker">gtk-color-picker</item>
<item stock="gtk-convert">gtk-convert</item>
<item stock="gtk-connect">gtk-connect</item>
<item stock="gtk-copy">gtk-copy</item>
<item stock="gtk-cut">gtk-cut</item>
<item stock="gtk-delete">gtk-delete</item>
<item stock="gtk-directory">gtk-directory</item>
<item stock="gtk-disconnect">gtk-disconnect</item>
```

```
<item stock="gtk-edit">gtk-edit</item>
<item stock="gtk-execute">gtk-execute</item>
<item stock="gtk-file">gtk-file</item>
<item stock="gtk-find">gtk-find</item>
<item stock="gtk-find-and-replace">gtk-find-and-replace</item>
<item stock="gtk-floppy">gtk-floppy</item>
<item stock="gtk-fullscreen">gtk-fullscreen</item>
<item stock="gtk-goto-bottom">gtk-goto-bottom</item>
<item stock="gtk-goto-first">gtk-goto-first</item>
<item stock="gtk-goto-last">gtk-goto-last</item>
<item stock="gtk-goto-top">gtk-goto-top</item>
<item stock="gtk-go-back">gtk-go-back</item>
<item stock="gtk-go-down">gtk-go-down</item>
<item stock="gtk-go-forward">gtk-go-forward</item>
<item stock="gtk-go-up">gtk-go-up</item>
<item stock="gtk-harddisk">gtk-harddisk</item>
<item stock="gtk-help">gtk-help</item>
<item stock="gtk-home">gtk-home</item>
<item stock="gtk-index">gtk-index</item>
<item stock="gtk-indent">gtk-indent</item>
<item stock="gtk-info">gtk-info</item>
<item stock="gtk-unindent">gtk-unindent</item>
<item stock="gtk-italic">gtk-italic</item>
<item stock="gtk-jump-to">gtk-jump-to</item>
<item stock="gtk-justify-center">gtk-justify-center</item>
<item stock="gtk-justify-fill">gtk-justify-fill</item>
<item stock="gtk-justify-left">gtk-justify-left</item>
<item stock="gtk-justify-right">gtk-justify-right</item>
<item stock="gtk-leave-fullscreen">gtk-leave-fullscreen</item>
<item stock="gtk-missing-image">gtk-missing-image</item>
<item stock="gtk-media-forward">gtk-media-forward</item>
<item stock="gtk-media-next">gtk-media-next</item>
<item stock="gtk-media-pause">gtk-media-pause</item>
<item stock="gtk-media-play">gtk-media-play</item>
<item stock="gtk-media-previous">gtk-media-previous</item>
<item stock="gtk-media-record">gtk-media-record</item>
<item stock="gtk-media-rewind">gtk-media-rewind</item>
<item stock="gtk-media-stop">gtk-media-stop</item>
<item stock="gtk-network">gtk-network</item>
<item stock="gtk-new">gtk-new</item>
<item stock="gtk-no">gtk-no</item>
<item stock="gtk-ok">gtk-ok</item>
<item stock="gtk-open">gtk-open</item>
<item stock="gtk-paste">gtk-paste</item>
<item stock="gtk-preferences">gtk-preferences</item>
<item stock="gtk-print">gtk-print</item>
<item stock="gtk-print-preview">gtk-print-preview</item>
<item stock="gtk-properties">gtk-properties</item>
<item stock="gtk-quit">gtk-quit</item>
<item stock="gtk-redo">gtk-redo</item>
<item stock="gtk-refresh">gtk-refresh</item>
<item stock="gtk-remove">gtk-remove</item>
```

```
<item stock="gtk-revert-to-saved">gtk-revert-to-saved</item>
<item stock="gtk-save">gtk-save</item>
<item stock="gtk-save-as">gtk-save-as</item>
<item stock="gtk-select-color">gtk-select-color</item>
<item stock="gtk-select-font">gtk-select-font</item>
<item stock="gtk-sort-ascending">gtk-sort-ascending</item>
<item stock="gtk-sort-descending">gtk-sort-descending</item>
<item stock="gtk-spell-check">gtk-spell-check</item>
<item stock="gtk-stop">gtk-stop</item>
<item stock="gtk-strikethrough">gtk-strikethrough</item>
<item stock="gtk-undelete">gtk-undelete</item>
<item stock="gtk-underline">gtk-underline</item>
<item stock="gtk-undo">gtk-undo</item>
<item stock="gtk-yes">gtk-yes</item>
<item stock="gtk-zoom-100">gtk-zoom-100</item>
<item stock="gtk-zoom-fit">gtk-zoom-fit</item>
<item stock="gtk-zoom-in">gtk-zoom-in</item>
<item stock="gtk-zoom-out">gtk-zoom-out</item>
</tree>
</frame>
<hbox>
<button cancel></button>
<button ok></button>
</hbox>
</vbox>
'

gtkdialog --program=MAIN_DIALOG
```

Icons

Entering the following in a terminal:

Using the following command works fine:

**yad --button=gtk-ok:0**

However, sometimes the icons don't show, for example:

**yad --button=gtk-color-picker:0**

To use the gtk-color-picker icon and many others you need to use the separator
but it needs to be escaped. Note the \ before the !

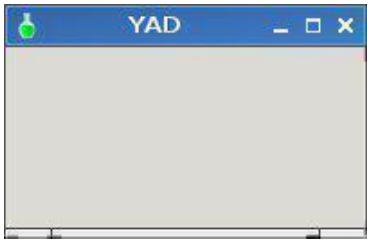
**yad --button=Colours\!gtk-color-picker:0**

When using the same commands in a script, escaping the ! is not required.

--no-buttons

As it says, this will create a dialog without any buttons.

**yad --no-buttons --width=200 --height=100**




I added the **--width** and **--height** commands or the dialog would have been very small.

It's actually the buttons on the default dialog that set the width.

Example:

**yad --no-buttons**

Not very attractive.

--no-markup

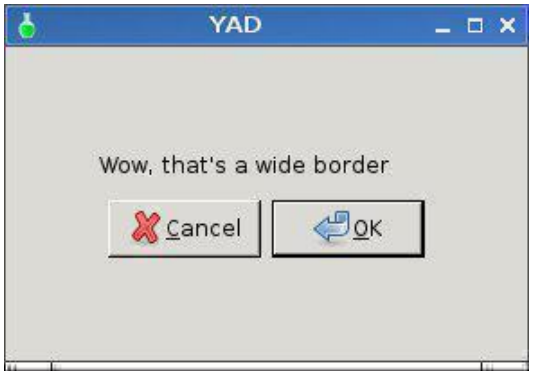
**--no-markup** Don't use pango markup language in dialog's text

Not sure what this means

--borders=NUMBER

This sets the width of the border around the dialog.

**yad --borders=100 --text="Wow, that's a wide border"**




--always-print-result

Haven't worked this one out yet.

--selectable-labels

This allows you to select the text from a dialog. The default in no select.

**yad --text="nah nah you can't select me"**

**yad --text="Ah ooh, now you can" --selectable-labels**


This means you can copy and paste from the dialog.

--sticky

Makes the GUI visible on all desktops.

--fixed

The **--fixed** command removed the ability to resize the dialog.

Note the resizing handles have disappeared.

**yad --fixed**


--on-top

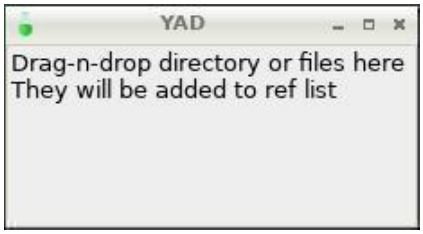
Places the dialog on top of other windows.

```
#!/bin/sh
yad --title="ON-TOP" \
--text="I'm a window that is always on top of (in the foreground of)
other windows, even if you select them, I won't disappear.
Specially handy for dragging stuff from other windows to here." \
--on-top
```


Here s an example of --on-top 'window above other windows always except when full screen such as video etc'.
Also an example of: --skip-taskbar This prevents the script showing in the task bar.

```
#!/bin/sh
yad --geometry=100x100 --on-top --text="Drag-n-drop directory or files here
```

**\nThey will be added to ref list"** --no-buttons --skip-taskbar --dnd --cmd echo $1 | sed 's/^.....//' >> /root/references.txt




--center

This is an easy way to position the dialog in the centre of your screen.

**yad --center**

--mouse

This will place the dialog at the position where your mouse cursor is.

**yad --mouse**

--undecorated

This command removed the title bar and borders from the dialog.

**yad --undecorated**


If you do **--no-buttons** and **--undecorated** like this:

**yad --no-buttons --undecorated**

You will have trouble closing the dialog unless you use a **--timeout=5**

**yad --no-buttons --undecorated --timeout=5 --text="Look ma no buttons and borders"**

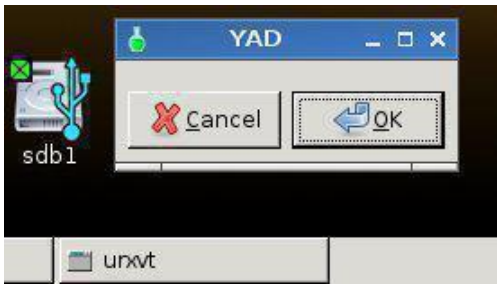
**yad --no-buttons --undecorated --timeout=5 --text="Look ma no buttons and borders"**


--skip-taskbar

As it says. It doesn't show the dialog in the task bar.

Yes, I know this may be confusing so here is a shot of my task bar with and without the
**--skip-taskbar** command.

**yad --skip-taskbar**




**yad**


--kill-parent

I think this kills the parent if it's been spawned.

More experimenting required.

--print-xid

--print-xid Print X Window Id to the stderr

More research required

Calendar options

This will display a calendar showing the current Day, Month and Year.

**yad --calendar**

It will also show the date selected so you can parse it to something else.

You can capture the date as a variable like this and parse to to another dialog:

```
#!/bin/sh
THEDATE=`yad --calendar`
yad --text="You chose $THEDATE"
```

--day=DAY

**yad --calendar --day=20** sets the 20th day of the current month.


--month=MONTH

**yad --calendar --month=2** sets the month of the current year and day.


--year=YEAR

**yad --calendar --year=2012** sets the month of the current year and day.

Houston we have a problem. The year switch doesn't appear to work in Fatdog.

**yad --calendar --year=2012**
Bring up the current date not 2012 as you would expect.

However this works:
**yad --calendar --year=2012 --month=2 --day=21**
--date-format=PATTERN

**yad --calendar** will produce the following output of the current day month and year:
**25/01/15**

**yad --calendar --date-format=%d%m%y**
Output will be: **250115**
--details=FILENAME

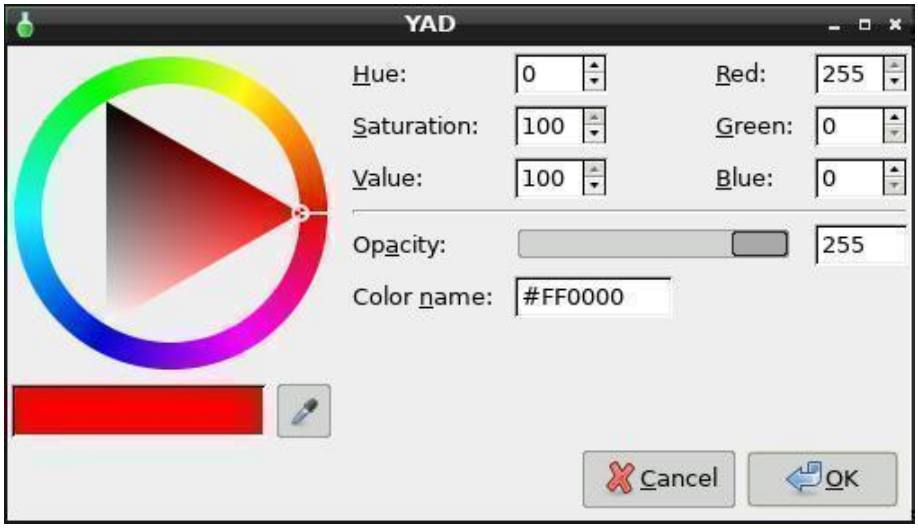
Not sure what this does.
Color selection options

**--fore and --back works only for --text-info dialog**

The --text-info command reads it's data from an external file.
--init-color=COLOR

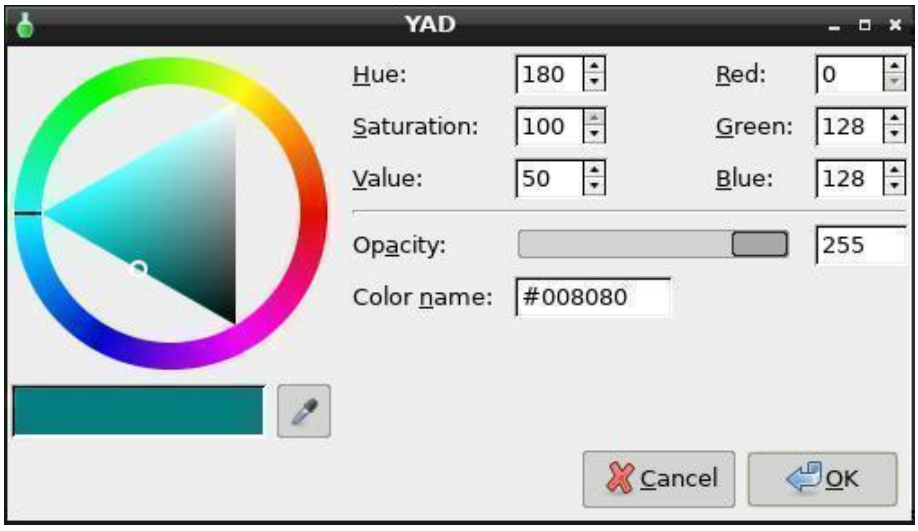
Initial colour

**yad --color --init-color=red**




or

**yad --color --init-color=#008080**




Output: #008080
--palette=FILENAME
--extra
Drag & Drop options

This will make the GUI able to Drag and Drop.

When you drop a file on the GUI it will output the filepath.

**yad --dnd**


After dropping a file on the dialog you get this:
file:///root/yad.jpg

Pipe it to sed to get a proper path:
**yad --dnd | sed 's/^.....//'**
/root/Desktop/tmp

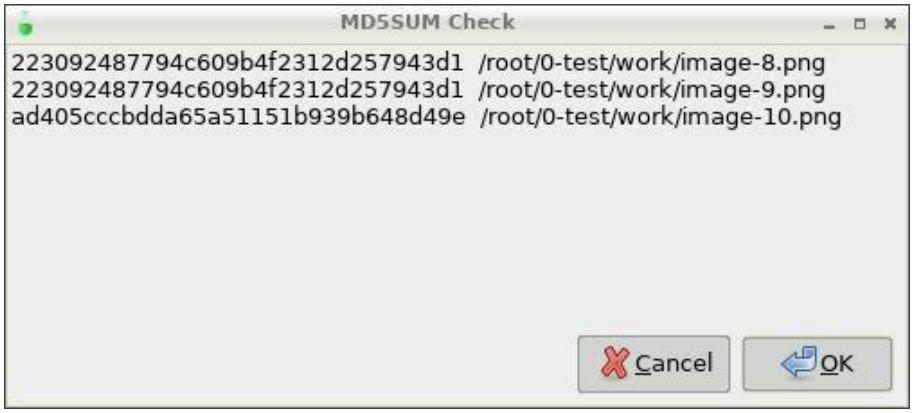
You can also drag and drop multiple files onto the gui.

Here is an example of a md5sum check.
Create the script below and make it executable.
Execute the script, drop files onto it and press OK.

```
#!/bin/sh
v=`yad --text="Drop your file(s) here:" --dnd | sed 's/^.....//'`
o=`md5sum $v`
yad --width=500 --height=200 --title="MD5SUM Check" --text="$o"
```

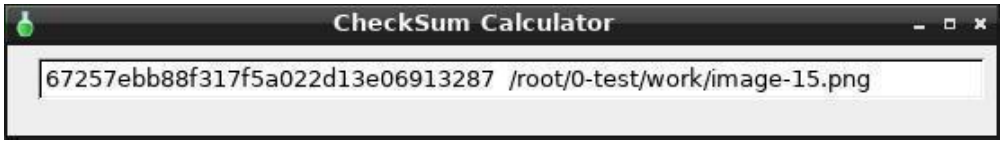
The first GUI looks like this:


After you drop the files and click Close you get:




Another method is to create the script below. Drag it to your desktop then drag and drop files onto the script. If you drop more than one file at a time you will have to close the top GUI to see the next one. The script is designed for single file drops.

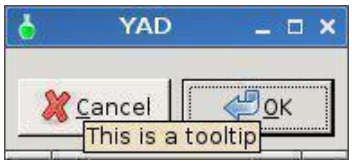
```
#!/bin/sh
killall yad
for i in "$@"; do
case "${i}" in
*)
o=`md5sum "${i}"`
yad --title="CheckSum Calculator" --form --field="" "$o" --no-buttons
--geometry=550x50+500+0
;;
esac
done
```




--tooltip

> You can use tooltips on this option by simply adding **--tooltip**. Adding this command alone is not enough. You see, **--tooltip** steals the text from the **--text** command so you need to do something like this:

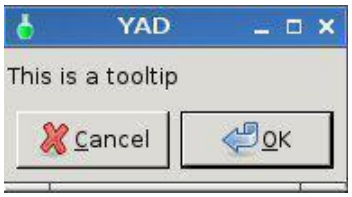
**yad --dnd --text "This is a tooltip" --tooltip**




You should also notice the text is no longer displayed on the dialog.

If you remove --tooltip from the command, EG:

**yad --dnd --text "This is a tooltip"**




The text has returned.

--command=CMD

This is a very nice feature.

This provides the ability to drop a file or files onto the dialog then call an application or a script and do something with it.

Lets assume we want to drop a few graphic files onto the dialog then open them in your default graphics viewer. Let's also assume you viewer is viewnior.

Copy and paste the following script to your text editor and make the file executable.

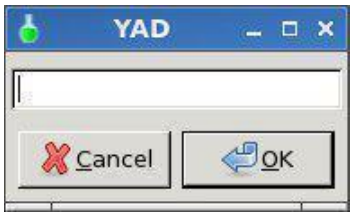
```
#!/bin/sh
ANSWER=`yad --text="Drag graphics files here" --dnd --tooltip`
viewnior $ANSWER
```

When you run this script, it will popup a dialog. If you hover your mouse over it you will see the tooltip. Now drag and drop one or many graphic files on the dialog.
When you click on the Ok button Viewnior will open and display your graphic files.

Text entry options

Show an entry dialog.

**yad --entry**




--entry-label=TEXT

Add a label to the entry dialog.

**yad --entry --entry-label="Label"**


You need to use both the **--entry** and **--entry-label=TEXT** commands.

--entry-text=TEXT

Now we will add to the previous example and add **--entry-text**.

**yad --entry --entry-label="Label" --entry-text="Type here"**


You can also add the **--text** command to the dialog and center it.

**yad --text-align=center --text="Hello" --entry --entry-label=Label --entry-text="Type here"**


When you press enter the "Type here" or whatever text you entered will be parsed. To capture the parsed input you can set the entire dialog to a variable like this:

**#!/bin/sh**

```
INPUTTEXT=`yad --text-align=center --text="Hello" --entry --entry-label=Label --entry-text="Type here"`
yad --text="You entered: $INPUTTEXT"
```

Markup

The text accepts markup the GUI height and width.

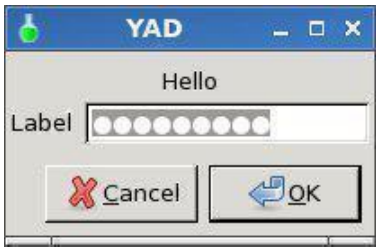
```
#!/bin/sh
yad --title="Golf Club" --height=200 --width=400 --text="<span foreground='blue'><b><big><big>Please enter your details:</big></big></b></span>" \
--form \
--field="<b><big><big>Golflink Number</big></big></b>" \
--field="<b><big><big>Score</big></big></b>" \
```


As you can see the text is now bigger.

--hide-text=TEXT

This command allows you to hide the text when you type. EG: Passwords or similar.

```
yad --text-align=center --text="Hello" --entry --entry-label=Label --entry-text="Type here" --hide-text
```




If you omit the **--entry-text="Type here"** the entry box will be blank.

```
yad --text-align=center --text="Hello" --entry --entry-label=Label --hide-text
```


You will also notice that you don't need to add any text to the **--hide-text** command.

--completion

The --completion command searches the data in the script and matches it against your input. Run the script and type b into the entry box. You will be given a choice of three words beginning with b.

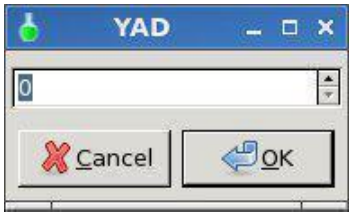
```
#!/bin/sh
yad --title="--completion" --completion \
--entry="Name:" "" \
"Abba" "Bark" "Bungy Boy" "Billy" "Charlie" "Delta" "Echo" | while read line; do
ENTRY=`echo $line | awk -F',' '{print $1}'`
echo $ENTRY
done
```


Now after typing b


Whatever you choose will become the output.

--numeric

```
yad --entry --numeric
```



Uses a spin button for text entry and won't allow character entry.

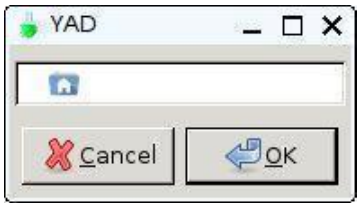
Entering data can be manually typed in.

There are 6 decimal points. EG: 100.000000
--editable

I haven't worked this one out yet.

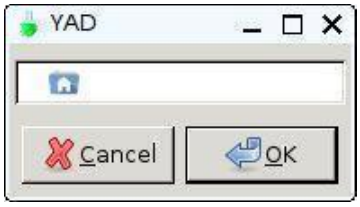
Can edit the text.
--licon=IMAGE

**yad --entry --licon=/usr/local/lib/X11/pixmaps/home48.png**



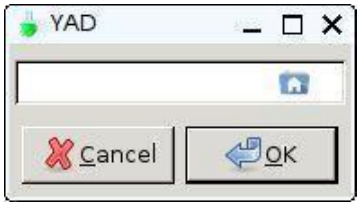

Notice the icon justified to the left of the entry box.
--licon-action=CMD

**yad --entry --licon=/usr/local/lib/X11/pixmaps/home48.png --licon-action=rox**

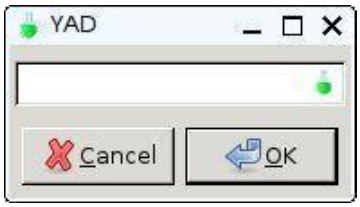



Click on the rox icon in the entry field to launch rox.
--ricon=IMAGE

**yad --entry --ricon=/usr/local/lib/X11/pixmaps/home48.png**

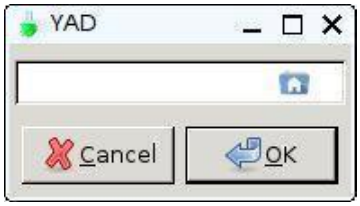



If the icon cannot be found the yad icon will be used.




Notice the icon justified to the right of the entry box.
--ricon-action=CMD

**yad --entry --ricon=/usr/local/lib/X11/pixmaps/home48.png --ricon-action=rox**

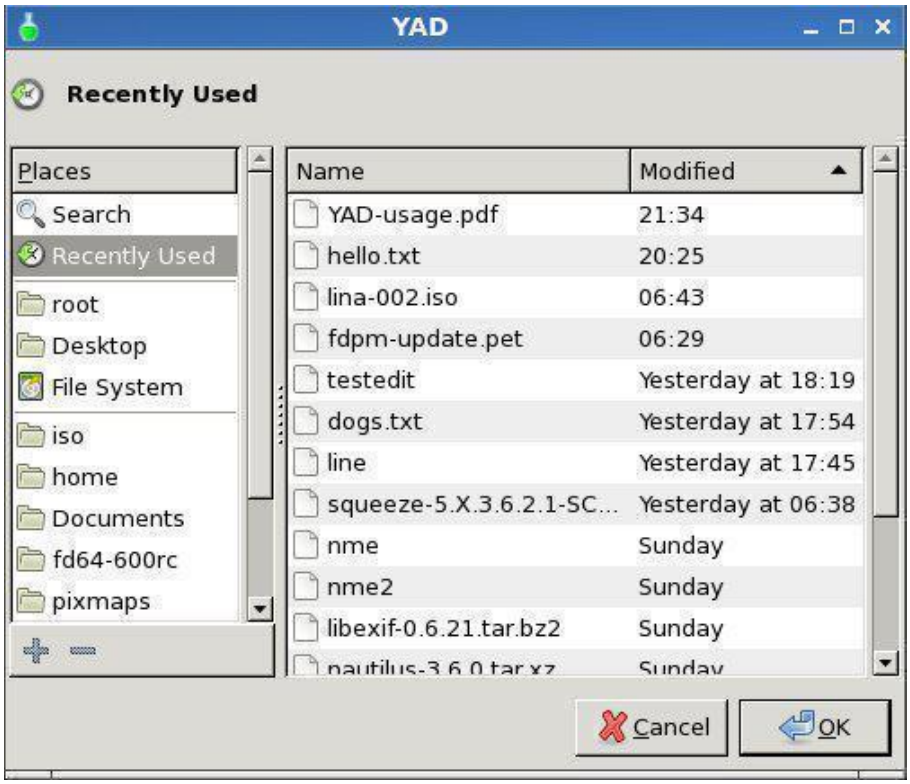



Click on the rox icon in the entry field to launch rox.
File selection options

**yad --file** will invoke the file select dialog


As you can see the dialog is a bit small to display text. Try increasing width and height.

**yad --width=500 --height=400 --file**

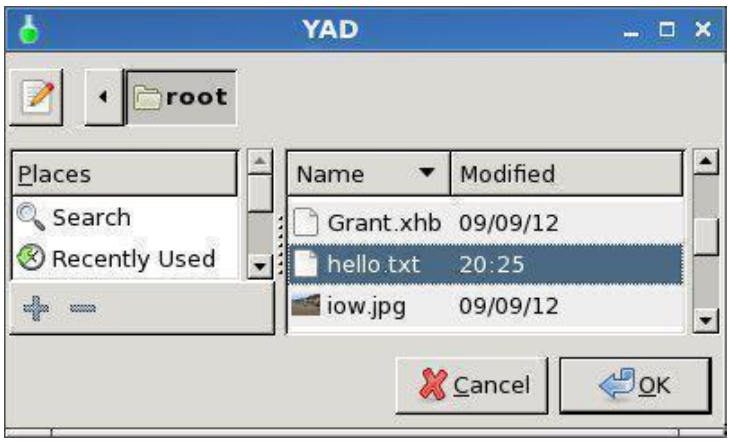



--filename=FILENAME

The **file selection options** need to be prefixed by yad --file EG:

**yad --file --filename=/root/hello.txt --width=400** (/root/hello.txt must exist)

This will select /root/hello.txt

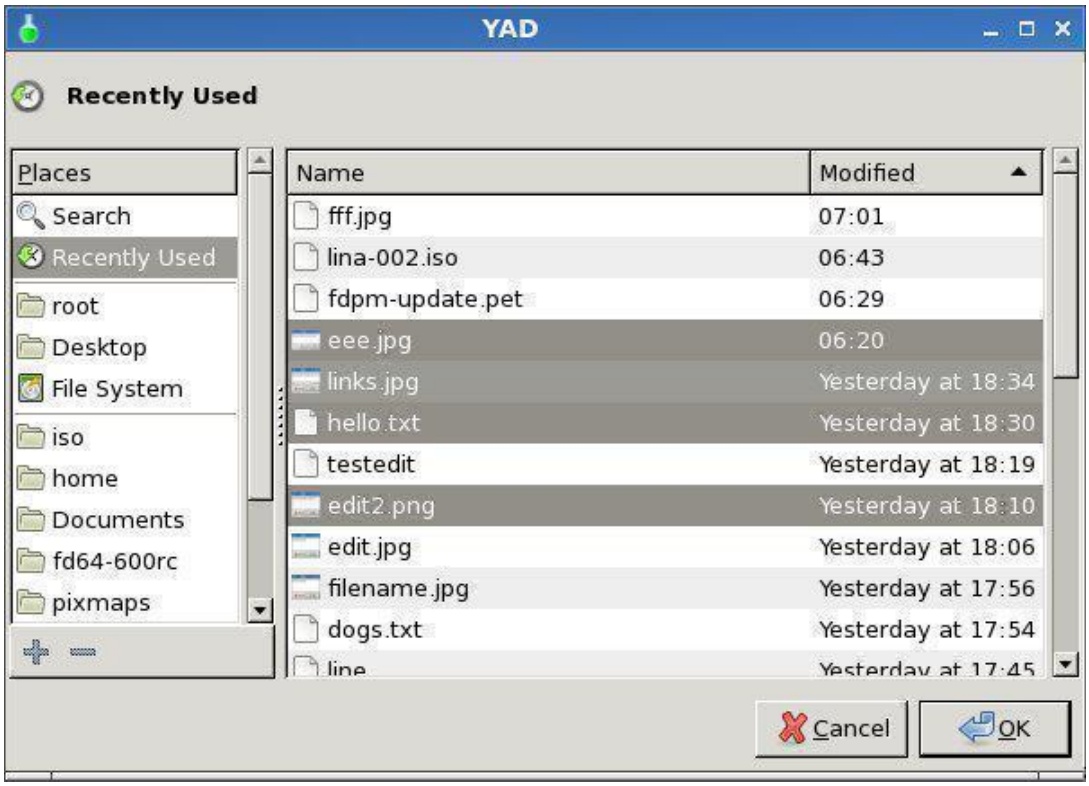



Notice the file **hello.txt** is already selected.

--multiple

This option will allow you to select multiple files at the same time.

**yad --file --multiple --width=600 --height=400**




Use the Ctrl or Shift keys when selecting.

It's all well and good being able to bring up a file select dialog but it needs to be able to parse the
information selected.

One way is to assign the file selection dialog to a variable like:

**CHOOSEFILE=`yad --file --width=600 --height=400`**

To see how it works try selecting a graphics file in a script like this:

```
#!/bin/sh
CHOOSEFILE=`yad --file --width=600 --height=400`
viewnior $CHOOSEFILE
```

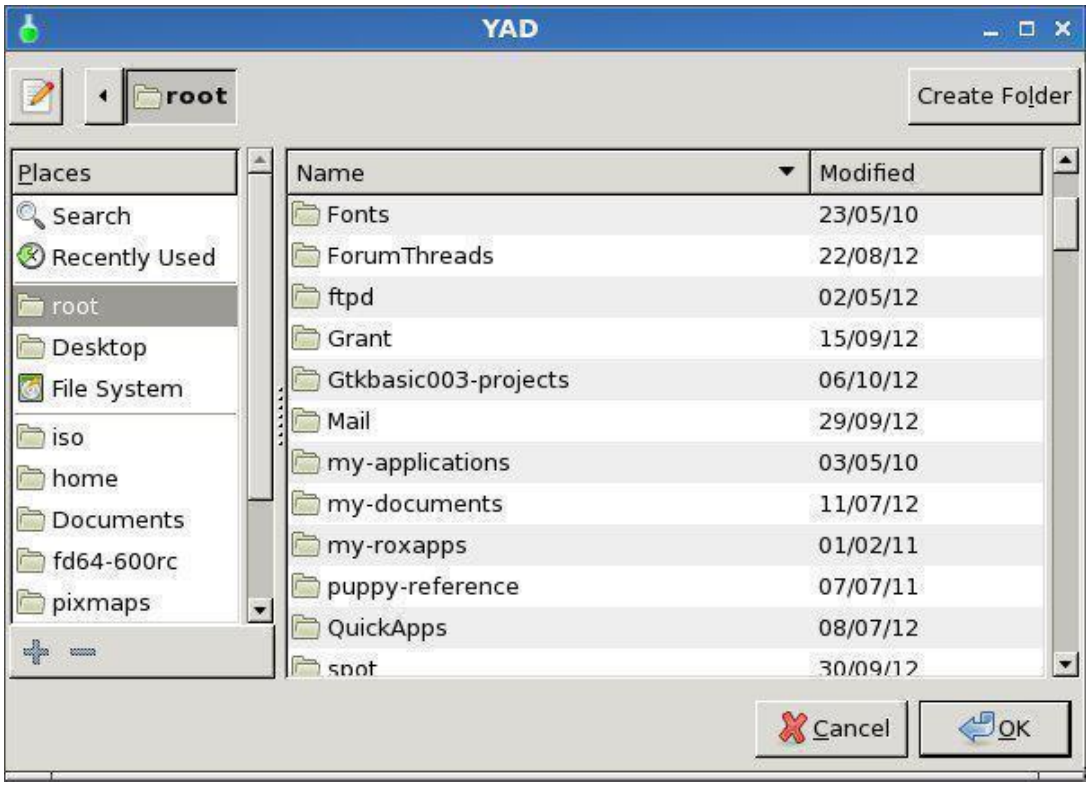
Make sure you make the script executable. In Rox, right click on the file, select Permission and
then click on the Yes button.

Note the little mark before yad ` and after 400` this is called a backtick, it's the key under the Esc key.

--directory

The **--directory** command is used to select a directory/folder.

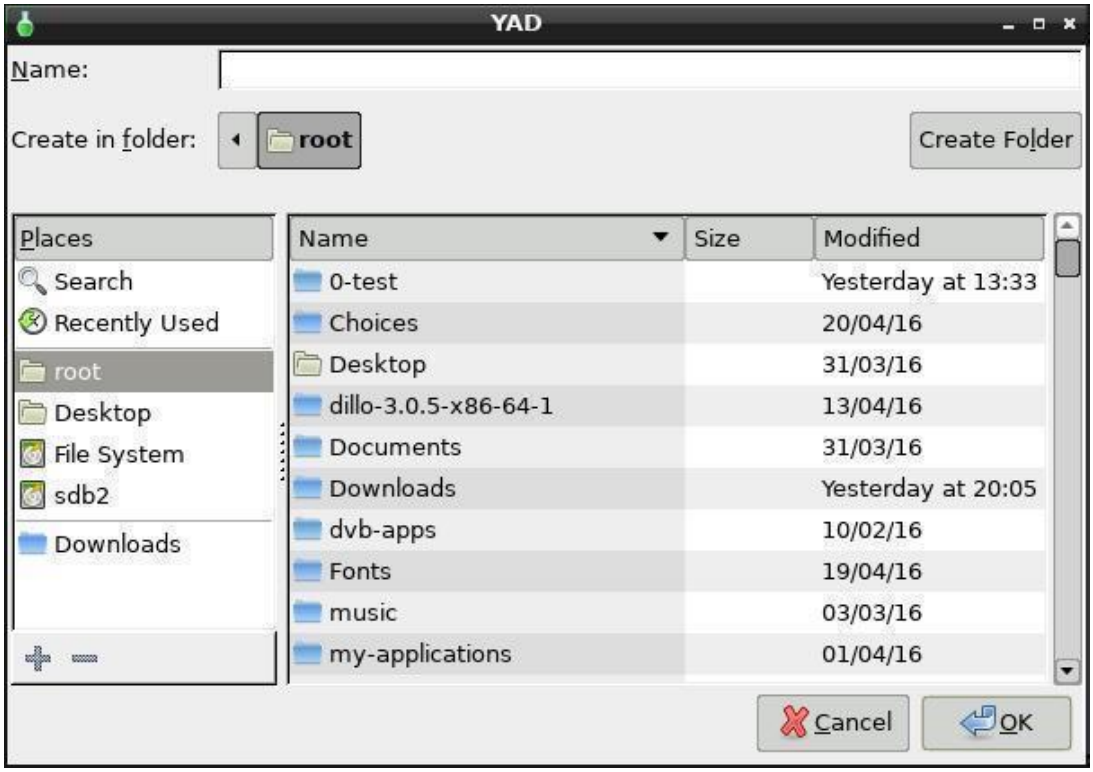
**yad --file --directory --width=600 --height=400**




Only directories will be displayed, not files.

--save

**yad --file --directory --save --width=600 --height=400**



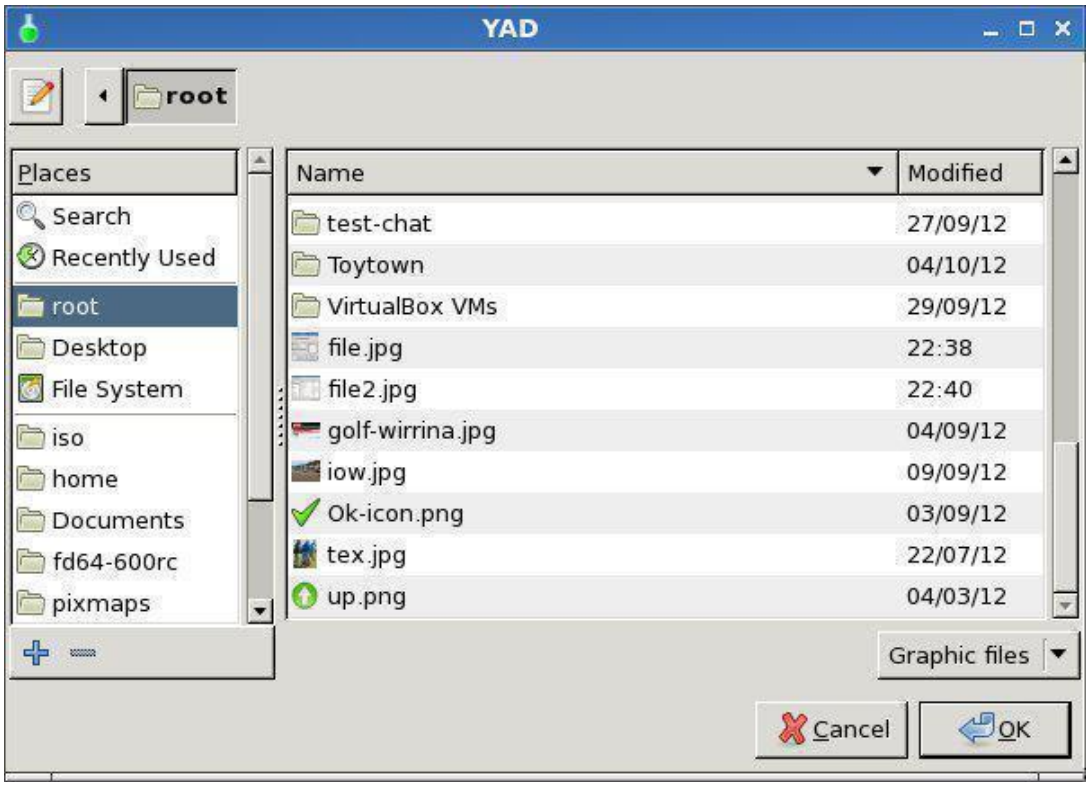
This provides a way to save a file. If you type hello.txt in the name field and click on the root
directory, then press OK the output will be: /root/hello.txt

--separator=SEPARATOR
--confirm-overwrite=[TEXT]
--file-filter=NAME | PATTERN1 PATTERN2 ...

The --file-filter command will filter the files.

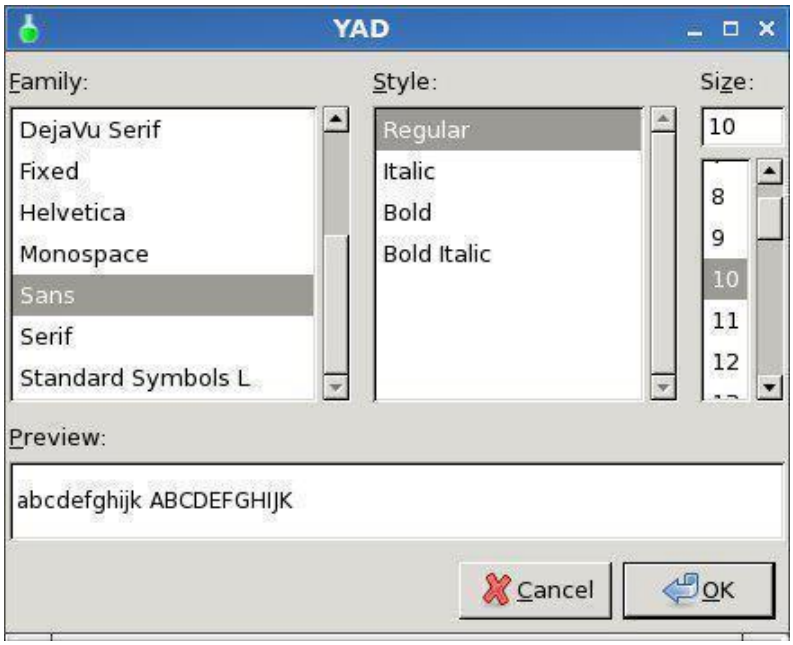
If you only want to display **jpg** and **png** files then:

**yad --width=600 --height=400 --file --file-filter "Graphic files | *.jpg *.png"**




Directories will also be displayed along with jpg and png files.
Font selection options

**yad --font** will invoke the font dialog.




Output in the terminal is:
**Sans 10**

If you were to select the Bold Style then:
**Sans Bold 10**

To pass the three, (Family, Style, Size) separately, you might need to do something like this:

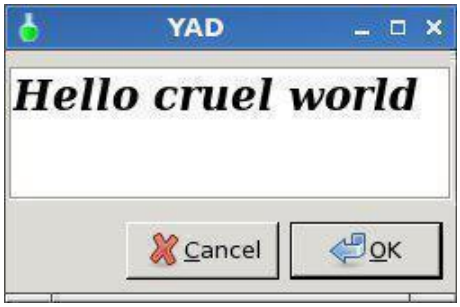
```
#!/bin/sh
yad --font | while read line; do
STL=`echo $line | grep -o "Bold\|Italic\|Oblique\|Condensed\|Medium"`
#Might need to add more styles
STYLE=`echo $STL`
SIZE=`echo $line | awk '{print $NF}'`

if [ "$STYLE" = "" ]; then
NAME=`echo $line | awk -F"$SIZE" '{print $1}'`
else
NAME=`echo $line | awk -F"$STYLE" '{print $1}'`
fi

echo $NAME $STYLE $SIZE

done
```

--fontname=FONTNAME

This command sets the font type.

You can also specify bold, italic, and size. Note the quotes.

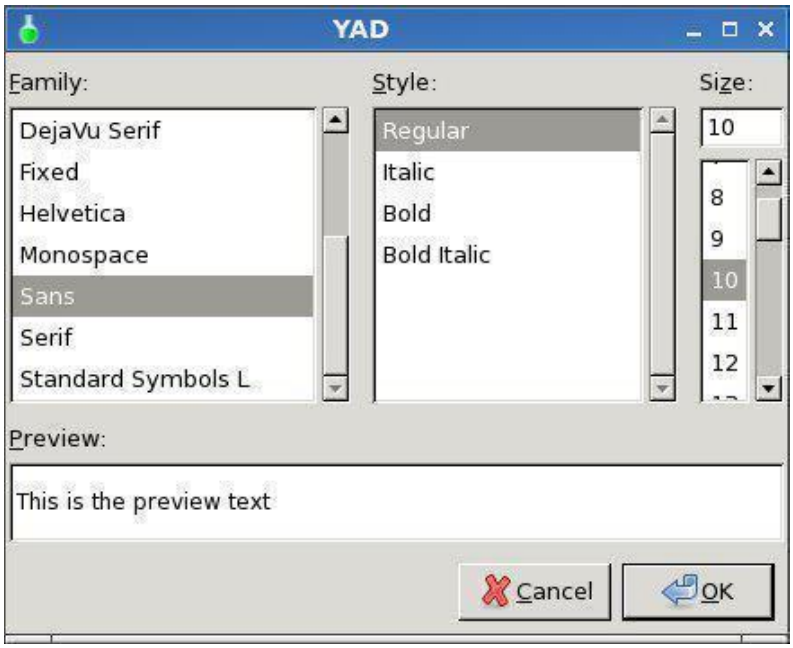
**yad --width=250 --text-info < hello.txt --fontname="Serif bold italic 20"**




--preview=TEXT

The **--preview** command allows you to set the preview text

**yad --font --preview "This is the preview text"**




The = symbol after preview is optional.

Form options

http://www.thelinuxrain.com/articles/multiple-item-data-entry-with-yad

http://rpm.pbone.net/index.php3/stat/45/idpl/17003956/numer/1/nazwa/yad

```
holidays=$(echo "Gold Coast,Bali,Phuket,Sydney,other")
yad --title="My YAD Test" --text="Please enter your details:" \
--image="/usr/share/icons/phone.png" \
--form --date-format="%-d %B %Y" --separator="," --item-separator="," \
--field="First Name" \
--field="Last Name" \
--field="Status":RO \
--field="Date of birth":DT \
--field="Last holiday":CBE \
```

--field="List your 3 favourite foods:":TXT \
"" "" "All round good guy" "Click calendar icon" "$holidays"


--field=LABEL[:TYPE]

Add field to form (TYPE - H, RO, NUM, CHK, CB, CBE, FL, SFL, MFL, DIR, CDIR, MDIR, FN,
DT, CLR, BTN, LBL or TXT)

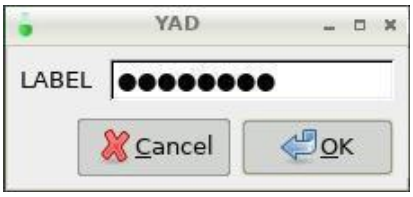
This is a simple form using the --field widget.

**yad --form --field=LABEL**


At the top of this page you will see many different field types.
Lets work our way through them one by one.
H - Hidden

The Type: H Hides the text, like a password.

**yad --form --field=LABEL:H**




When you click on the OK button the text is revealed in the CLI (Command Line Interface).

smokey01|

Notice the | (pipe) at the end of smokey01, this is called a seperator. More about thas later.

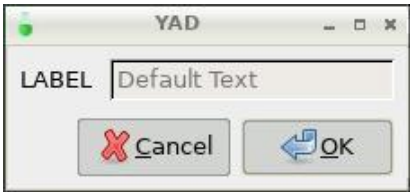
RO - Read Only

You might have guessed Type:RO mean Read Only. But if you run the following command:

**yad --form --field=LABEL:RO**

You won't see any text so we need to add some default text like this:

**yad --form --field=LABEL:RO "Default Text"**




Now you can see some text but it's greyed out.

NUM - Number

Type:NUM is for Numbers. Only numbers can be entered into this field.

**yad --form --field=LABEL:NUM**


Notice it has a spin box on the right hand side. This can be used to increase or decrease the value.
To make this a bit more useful we can set a default number, lets make it 10.

**yad --form --field=LABEL:NUM "10"**

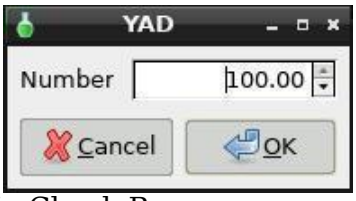

To control a Minimum and Maximum number, say 0.0 - 20.0:

**yad --form --field="Number:NUM" 0\!0..20\!1\!1**

**Notice the '\' escape characters. These are not required in a script but are from the CLI.**
**I will provide a script example at the end of the form options.**


I hear you ask, what about fractions. Lets set the Minimum to 0.0 and the Maximum to 100.0 with 0.5 increments.

**yad --form --field="Number:NUM" 0\!0..100\!0.05\!2**




CHK - Check Box

Type CHK places a check box in the form.

yad --form --field="Number:CHK"


Checked returns True, Unchecked returns False.

CB - Combo Box

Field Type CB refers to Combo Box. **Once again note the escape '\' characters.**

**yad --form --field="ComboBox:CB" One\!Two\!Three**


CBE - Combo Box Editable

Field Type CBE means the Combo Box is Editable. In other words you can add data on the fly by typing.

**yad --form --field="ComboBox:CBE" One\!Two\!Three**


You are not restricted to the three choices.

When you press OK in the above example
More Stuff| is returned.

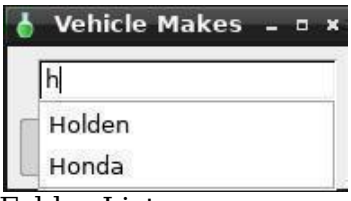
CE - Entry with Completion

The field Type CE completes a string from matching data. In the example below
There are a number of vehicle makes. If you type h in the entry box you will
be given a choice of vehicles listed that begin with h EG: Holden and Honda.

Run the script, you will get the idea.

```
#!/bin/sh
yad --title="Vehicle Makes" --form --field ':CE'
'Holden!Ford!Toyota!Honda!Fiat!Mitsubishi!Nissan!Aston
Martin!Rolls Royce!Jaguar!Renault!Bentley!Citroen'
```


Then type in h and you're given two choices.




FL - Folder List

Field type FL means Folder List.

**yad --form --field="Folder List:FL"**

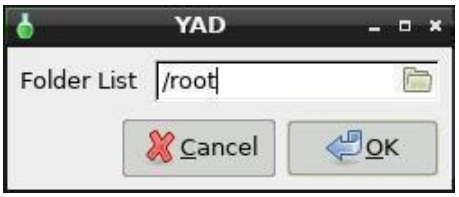

Notice (None) is displayed, this is because a path was not specified. Lets add a path to /root:

**yad --form --field="Folder List:FL" /root**


SFL - Select File List

Field Type SFL means select a file from a list.

**yad --form --field="Folder List:SFL" /root**




Now you can drill down through the folders, starting at /root, and select a file. The output will be the full /path/filename|

MFL - Multiple File List

Field Type MFL allows you to select multiple files. You need to hold the CTRL key while left mouse clicking to do so.

**yad --form --field="Folder List:MFL" /root**


The dialog looks the same as SFL however, when you select multiple files the /path/filenames are separated by a '!'

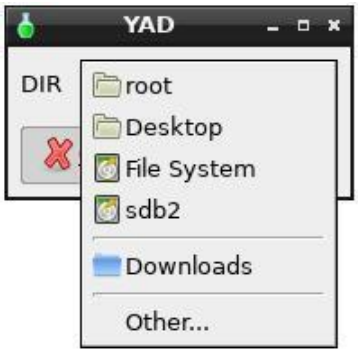
**/root/cruise0002.jpg!/root/cruise0003.jpg!/root/cruise0004.jpg|**

As you can see I selected three files.

DIR - Directory Structure

Field Type DIR display a basic directory structure.

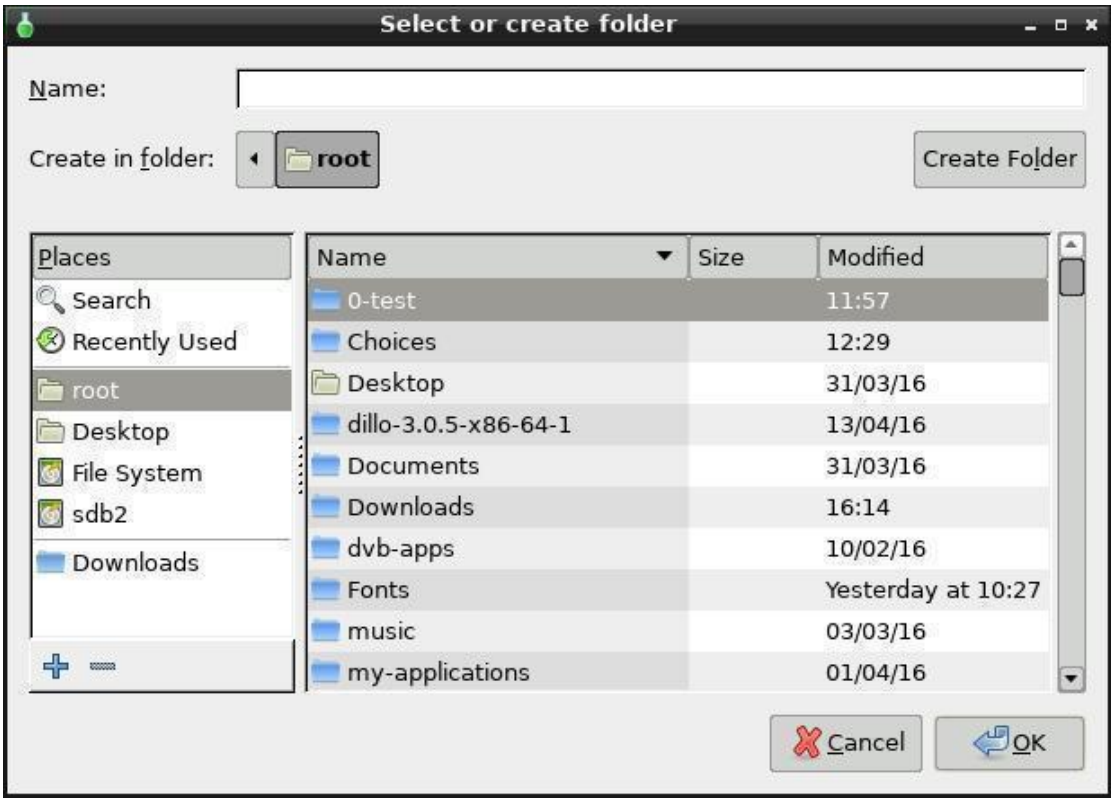
**yad --form --field="Folder List:DIR" /root**




CDIR - Select/Create Folders

Field Type CDIR is used to select or create folders.

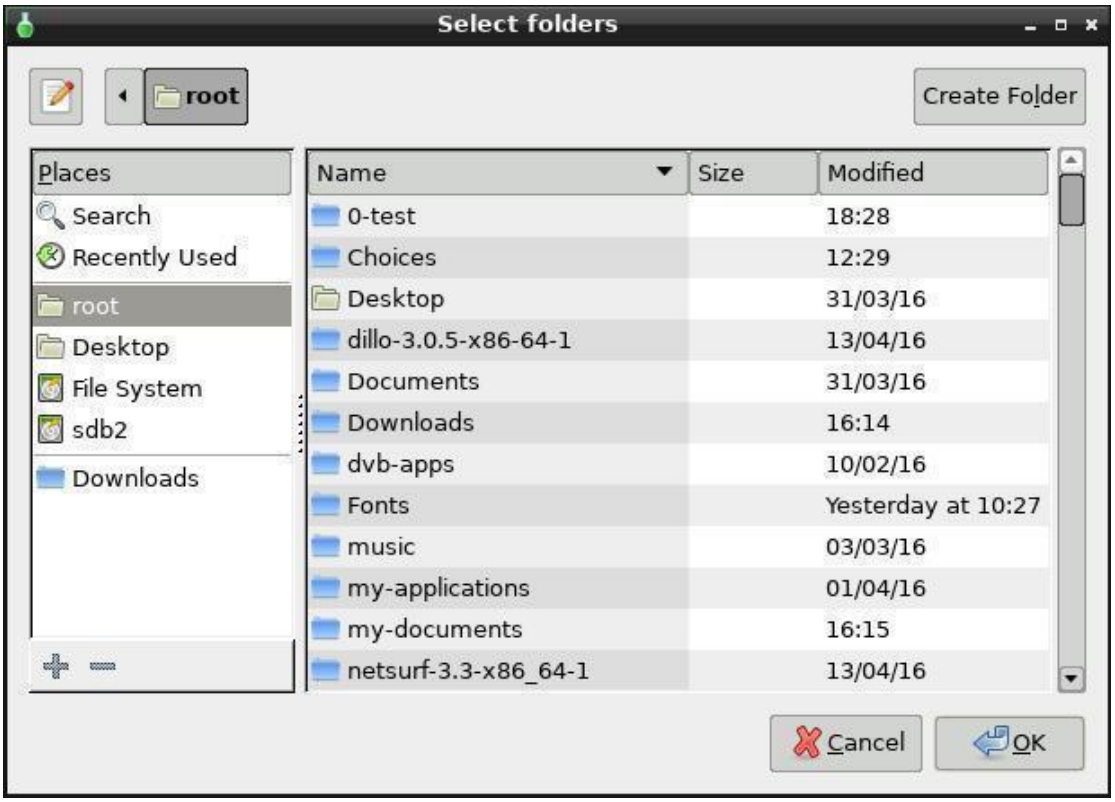
**yad --form --field="CDIR:CDIR" /root**




MDIR - Select Folders

Field MDIR is used to select folders only.

**yad --form --field="MDIR:MDIR" /root**

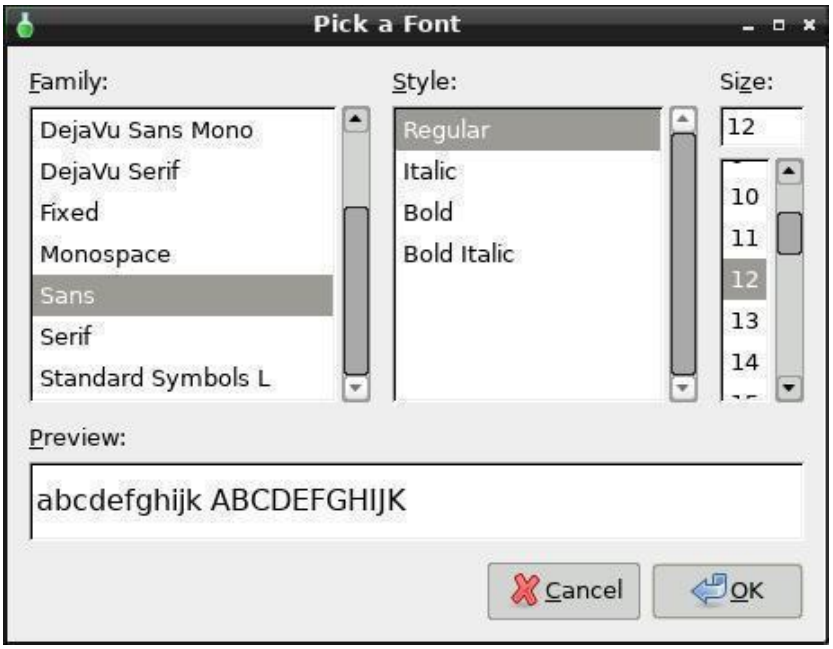



FN - Font Dialog

Field FN is used to display a Font Picker.

**yad --form --field="Font::FN"**


When you click on the Font Button:




**Tip: Did you notice "Font::FN" in the command?**

The first colon is added to the Label and the second defines the field type.

DT - Date Picker

Field DT provides a Date Picker

**yad --form --field="Date::DT"**


Notice the little Calendar in the right of the field.
When you click on it, you get a Calendar.

Select the date and click on OK. The following is returned:
20/04/16|

CLR - Color Picker

Field CLR provides a colour Picker.

**yad --form --field="Color::CLR"**

You can set a default colour by adding the Hex code:

y**ad --form --field="Color::CLR" "#F507D0"**

Click the Button to open the Colour Selector.

To make a selection you must click on the triangle and press enter.
Clicking on the wheel just moves the triangle.

Tip**: In the Color name field you can type recognised names like teal to select a colour.**
**There are hundreds defined so this may be an easier way to choose your colour.**

BTN - Field Button

Field BTN is for making Buttons.

**yad --form --field="Button::BTN"**


Now a button wouldn't be much use if it didn't
do something. Now we add a command so it
will open your default image viewer.

**yad --form --field="Image Viewer::BTN" defaultimageviewer**

LBL or TXT

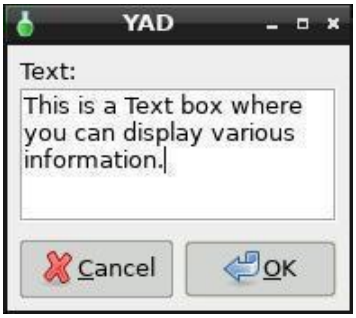
Field Type LBL means Label and TXT means Text.

**Tip: Whatever you put inside the quotes before the colon becomes the Label.**

**yad --form --field="This is a Label Maaate:LBL"**


Something similar with the TXT type.

**yad --form --field="Text::TXT" "This is a Text box where you can display various information."**



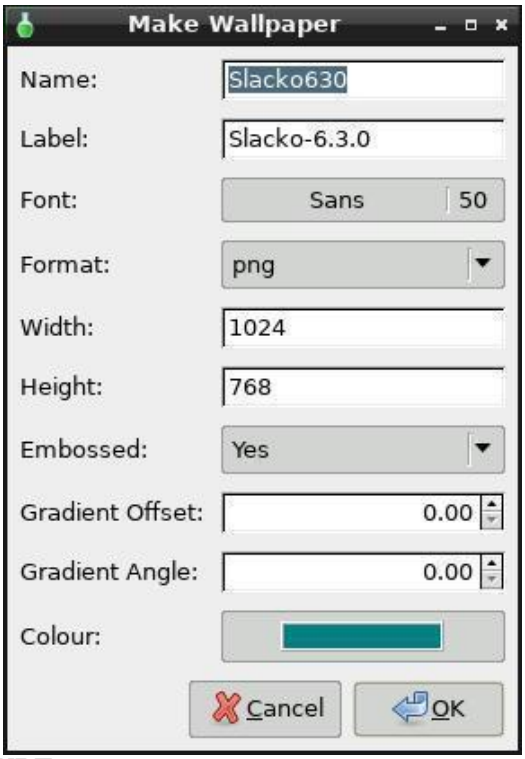

Form Example

This code was written as a GUI for 01micko's wonderful little CLI wallpaper maker.
The Code can be obtained here.

```
#!/bin/sh
#Written by Smokey01
#20 April 2016
#Requires YAD and 01micko's mkwallpaper
yad --title="Make Wallpaper" --form --separator="," \
--field="Name:" "Slacko630" \
--field="Label:" "Slacko-6.3.0" \
--field="Font::FN" "Sans 50" \
--field="Format::CB" "png!svg" \
--field="Width:" "1024" \
--field="Height:" "768" \
--field="Embossed::CB" "Yes!No" \
--field="Gradient Offset::NUM" 0!0..1!0.05!2 \
--field="Gradient Angle::NUM" 0!0..20!0.05!2 \
--field="Colour::CLR" "#008080" \
"" "" "" "" "" "" "" "" "" "" | while read line; do
IMAGENAME=`echo $line | awk -F',' '{print $1}'`
LABELNAME=`echo $line | awk -F',' '{print $2}'`
FONT=`echo $line | awk -F',' '{print $3}'`
FORMAT=`echo $line | awk -F',' '{print $4}'`
WIDTH=`echo $line | awk -F',' '{print $5}'`
HEIGHT=`echo $line | awk -F',' '{print $6}'`
EMBOSSED=`echo $line | awk -F',' '{print $7}'`
```

```
OFFSET=`echo $line | awk -F',' '{print $8}'`
ANGLE=`echo $line | awk -F',' '{print $9}'`
COLOUR=`echo $line | awk -F',' '{print $10}'`
echo $NAME $LABEL $FONT $FORMAT $WIDTH $HEIGHT $EMBOSSED
$OFFSET $ANGLE $COLOUR

# Convert the colour string xxxxxx to xx xx xx RGB
multi="0.003906"
red=`echo $COLOUR | cut -c2-3`
green=`echo $COLOUR | cut -c4-5`
blue=`echo $COLOUR | cut -c6-7`

# Convert colour string to decimal
fred="$((16#$red))"
fgreen="$((16#$green))"
fblue="$((16#$blue))"

# Scale the decimal numbers to 01micko's range
r=$(echo "$fred * $multi" | bc)
g=$(echo "$fgreen * $multi" | bc)
b=$(echo "$fblue * $multi" | bc)

# Separate font type from size
FONTY=`echo $FONT | awk '{ $NF = ""; print $0}'`
SIZEY=`echo $FONT | rev | cut -d' ' -f1 | rev`

# Run mkwallpaper, 01micko's cli application
mkwallpaper -n "$IMAGENAME" -l "$LABELNAME" -f "$FONTY" -p
$FORMAT -x $WIDTH -y $HEIGHT -s $SIZEY -k $EMBOSSED -o
"$OFFSET" -z "$r $g $b" -a $ANGLE

# Fixed a minor bug in the SVG format. Change pt to px to make sizing
work properly .
sed -i 's/pt/px/g' /usr/share/backgrounds/"$IMAGENAME.svg"

# Display wallpaper
defaultimageviewer /usr/share/backgrounds/$IMAGENAME.$FORMAT
done
```

This is what it looks like.




--align=TYPE

Set alignment of fields labels (left, center or right)

```
#!/bin/sh
yad --form --align=right \
--field="Label Aligned Right":LBL \
```


Label aligned right. Left and Center is also possible.

--columns=NUMBER

Set number of columns in form.

```
#!/bin/sh
yad --form --columns=2 \
--field="Firstname:" "Billy" \
--field="Age:" "21" \
--field="Lastname:" "Bloggs" \
--field="Sex::"CB "Male!Female" \
```


Notice the order of the fields counting from top to bottom. The first two fields are on the left while 3rd and 4th fields are on the right.

--separator=SEPARATOR

The default --separator is the | symbol, also known as a pipe.

```
#!/bin/sh
yad --form \
--field="Firstname:" "Billy" \
--field="Lastname:" "Bloggs" \
```

When you run the above script in a terminal you get this:


Then click OK you get this Output:
Billy|Bloggs|

Notice the pipes after Billy and Bloggs. This may be fine but what if you wanted spaces between Billy and Bloggs.

This is where you use the --separator command after the --form command. Let's add --separator=" " to the original script.

```
#!/bin/sh
yad --form --separator=" " \
--field="Firstname:" "Billy" \
--field="Lastname:" "Bloggs" \
```

The GUI looks the same but look at the output when OK is pressed.


Output: Billy Bloggs

--item-separator=SEPARATOR

Set the separator character for combobox or scale data.

You may have noticed on the --columns command I used a CB (Combo Box)

We want to add a few items to the CB so we need to use a separator character to separate them. The default character is the exclamation mark !

```
#!/bin/sh
yad --form \
--field="Sex::"CB "Male!Female!Other" \
```


Output: Make|

Now lets be bold and change both default separators, the --separator and --item-separator.

```
#!/bin/sh
yad --form --separator=" " --item-separator="," \
--field="Sex::"CB "Male,Female,Other" --item-separator="," \
```

The GUI looks the same but the CB choices are now separated by a comma instead of an exlamation mark and the pipe has disappeared from the end of the output. I selected Other and clicked on OK.

Output: Other

--date-format=PATTERN

Set the format for the returned date.

Run this script in a terminal:

```
#!/bin/sh
```

```
yad --form --separator=" " --date-format="%-d %B %Y" \
--field="Date:":DT \
```


Click on the little calendar, choose a date, click OK
then click OK again.

Output: 26 April 2016

Here are some date and time format codes in alphabetical order:

%% a literal %
%a locale's abbreviated weekday name (Sun..Sat)
%A locale's full weekday name, variable length (Sunday..Saturday)
%b locale's abbreviated month name (Jan..Dec)
%B locale's full month name, variable length (January..December)
%c locale's date and time (Sat Nov 04 12:02:33 EST 1989)
%d day of month (01..31)
%D date (mm/dd/yy)
%e day of month, blank padded ( 1..31)
%h same as %b, locale's abbreviated month name (Jan..Dec)
%H hour :24 hour(00..23)
%I hour :12 hour(01..12)
%j day of year (001..366)
%k hour :24 hour(00..23)
%l hour :12 hour(01..12)
%m month (01..12)
%M minute (00..59)
%n a newline
%p locale's AM or PM
%r Time, 12-hour (hh:mm:ss [AP]M)
%s Seconds since 1970-01-01 00:00:00, (a GNU extension)
Note that this value is defined by the localtime system
call. It isn't changed by the '--date' option.
%S second (00..60)
%t a horizontal tab
%T Time, 24-hour (hh:mm:ss)
%U Week number of year with Sunday as first day of week (00..53)
%V Week number of year with Monday as first day of week (01..53)
If the week containing January 1 has four or
more days in the new year, then it is considered week 1;
otherwise, it is week 53 of the previous year, and the next week
is week 1. Similar to ISO 8601 (but not 100% compliant.)

%w day of week (0..6); 0 represents Sunday
%W week number of year with Monday as first day of week (00..53)
%x locale's date representation (mm/dd/yy)

%X locale's time representation (%H:%M:%S)
%y last two digits of year (00..99)
%Y year (1970...)
%z RFC-822 style numeric timezone (-0500) (a nonstandard extension)
This value reflects the current time zone.
Is not changed by the --date option.
%Z Time offset from UTC (-07) This generally consists of Time Zone+DST
Is not changed by the --date option.

Icons box options

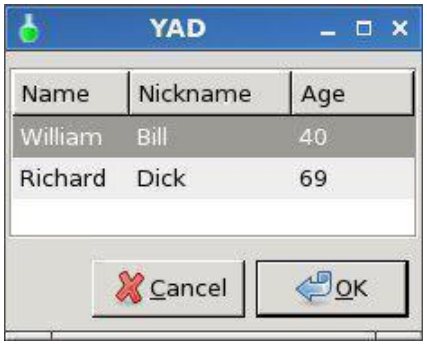
--read-dir=DIRECTORY
--compact
--generic
--stdin
--item-width
--term=PATTERN
Use specified pattern for launch command in terminal (default: xterm -e %s)

--sort-by-name
--descend

List options

You can do some very cool things with the --list command.

**yad --list --column=Name --column=Nickname --column=Age William Bill 40 Richard Dick 69**




Lets have a look at a really simple but useful shopping List example.

First create a text file called **/root/shopping.list**

In this list type or copy the following data:

true
milk
false
bread
false
eggs
false
butter
false

tomato soupg
false
flour
false
steak
false
sausages
false
carrots
true
beans

Here is the yad program that reads the file and lets you select the items.

**yad --height=300 --list --checklist --column=Buy --column=Item < /root/shopping.list**


This is good but maybe you need to print the list so you can take it to the shop.

**yad --button=gtk-print:0 --button=gtk-cancel:0 --height=300 --list --checklist --column=Buy --column=Item < /root/shopping.list > /root/printlist --print-column=2 --separator= && yad --print --add-preview --filename=/root/printlist**

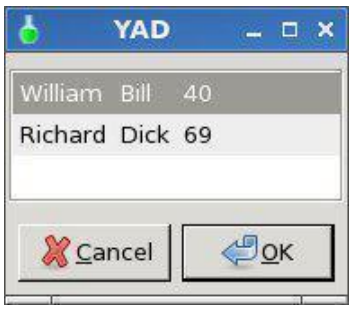

As you can see the OK button has been replaced with a Print button.

When you click on print, the Print dialog is invoked. You can Print Preview then Print the list if you wish.
--no-headers

Don't show column headers.

You need to use **--column** with the **--list** command but you don't have to display the headers.

**yad --list --no-headers --column=Name --column=Nickname --column=Age William Bill 40 Richard Dick 69**




--column=COLUMN[:TYPE]

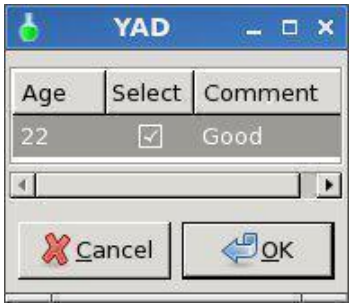
Set the column header (TYPE - TEXT, NUM, FLT, CHK, IMG or TIP)

--column=STRING[:TYPE] Set the column header. TYPE may be:

TEXT type is default (default)
NUM for integers
FLT for double values.
CHK checkboxes are a boolean columns.

RD radio toggle are a boolean columns.
IMG may be path to image or icon name from currnet GTK+ icon theme.Size of icons may be set in config file. Image field prints as empty value.
HD type means a hidden column. Such columns are not displayes in the list, only in output.
TIP is used for define tooltip column.

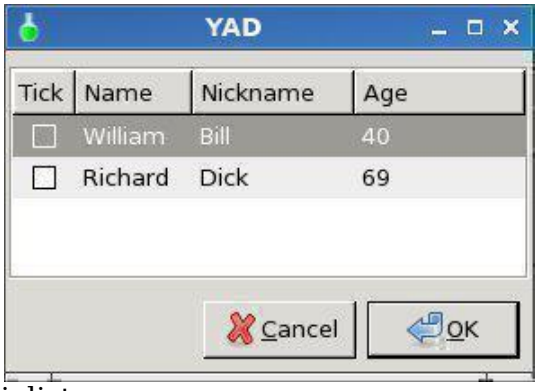
**yad --list --column=Age:num --column=Select:chk --column=Comment:text 22 true Good**




--checklist

Use check boxes for first column

**yad --checklist --list --column=Tick --column=Name --column=Nickname --column=Age false William Bill 40 false Richard Dick 69**




--radiolist

Alias to checklist (deprecated)

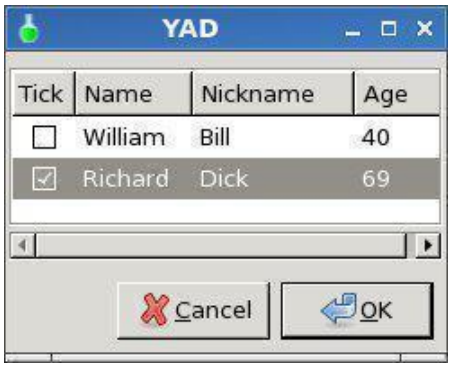
No longer used. --radiolist will use --checklist.

**yad --radiolist --list --column=Tick --column=Name --column=Nickname --column=Age false William Bill 40 false Richard Dick 69**


--separator=SEPARATOR

Set output separator character. The default is "|"

**yad --checklist --list --column=Tick --column=Name --column=Nickname --column=Age false William Bill 40 false Richard Dick 69**




When you select Richard and click on OK, this is the output you get:

**TRUE|Richard|Dick|69**

Now if you add --separator=, to the command list:

**yad --checklist --list --separator=, --column=Tick --column=Name --column=Nickname --column=Age false William Bill 40 false Richard Dick 69**

Your output is:
**TRUE,Richard,Dick,69,**

--multiple

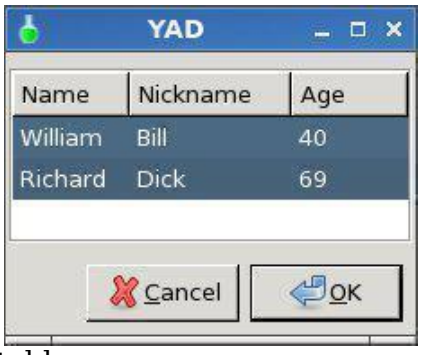
Allow multiple rows to be selected

When using the **--checklist** command you can select multiple rows. If you don't want a checklist then you need to use the **--multiple** command to select multiple rows.

Note: you also need to change the columns from 4 to 3 from the previous example and remove the false data.

**yad --list --multiple --column=Name --column=Nickname --column=Age William Bill 40 Richard Dick 69**

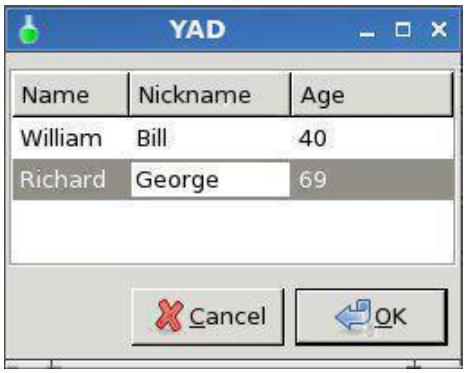
You do need to hold the shift or Ctrl key when selecting the row.




--editable

Allow changes to text.

**yad --list --editable --column=Name --column=Nickname --column=Age William Bill 40 Richard Dick 69**




You can edit each of the fields. When you click on OK the changes will be parsed.

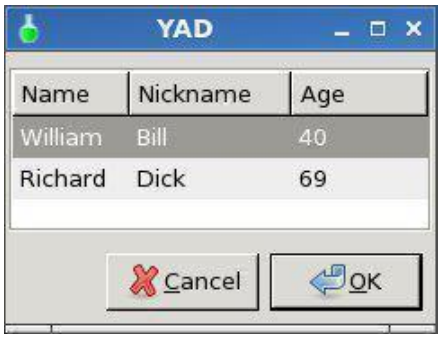
**Richard|George|69|**

--print-all

Print all data from list

**yad --list --print-all --column=Name --column=Nickname --column=Age William Bill 40 Richard Dick 69**

The **--print-all** command will parse all rows although I have only selected one.




**William|Bill|40|**
**Richard|Dick|69|**

--ellipsize=TYPE

Set ellipsize mode for text columns (TYPE - NONE, START, MIDDLE or END)

yad --list --ellipsize=end --column=Name --column=Nickname --column=Age
William Bill 40 Richard Dick 69

--print-column=NUMBER

Print a specific column. By default or if 0 is specified will be printed all columns

--hide-column=NUMBER

Hide a specific column

--expand-column=NUMBER

Set the column expandable by default. 0 sets all columns expandable

--search-column=NUMBER

Set the quick search column. Default is first column. Set it to 0 for disable searching

--limit=NUMBER

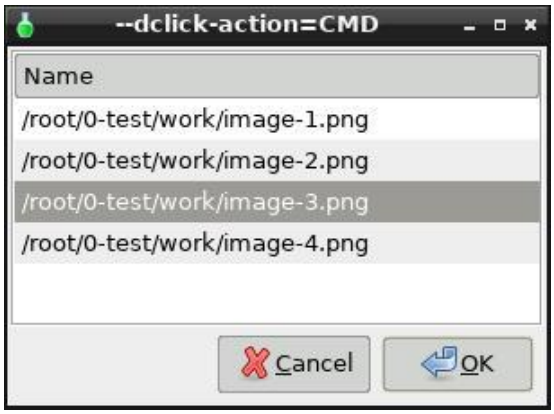
Set the limit of rows in list

--dclick-action=CMD

Set double-click action.

This command will allow you to double click on a list item and have it perform an action.

**yad --title="--dclick-action=CMD" --width=300 --height=200 --separator=" " --list --dclick-action="viewnior" --column=Name /root/0-test/work/image-1.png /root/0-test/work/image-2.png /root/0-test/work/image-3.png /root/0-test/work/image-4.png**




**Output: /root/0-test/work/image-3.png**

Rather than using viewnior (image viewer), Rox would be a btter choice as it understands
mime-type and will open the appropriate application for the data file.

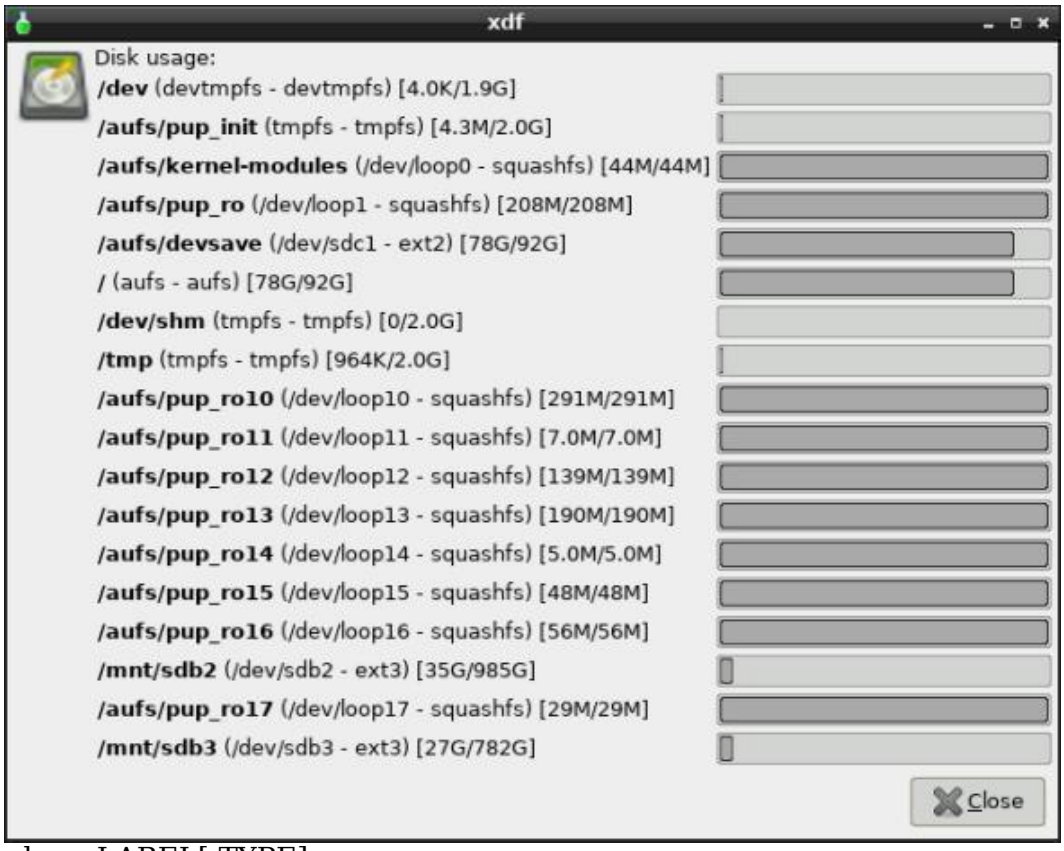
--regex-search

Use regex in search

Multi progress bars options

Here is an excellent example of multiple progress bars. It will only show the drives that are
currently mounted.

```
eval exec yad --title="xdf" --image=drive-harddisk --text="Disk\ usage:"
--buttons-layout=end --width=650 --multi-progress \
$(df -hT $1 | tail -n +2 | awk '{printf "--bar=\"<b>%s</b> (%s - %s)
[%s/%s]\" %s ", $7, $1, $2, $4, $3, $6}')
```




--bar=LABEL[:TYPE]

Add the progress bar (TYPE - NORM, RTL or PULSE)

--vertical

Show vertical bars

Notification icon options

--command=CMD

Set left-click action

--listen

Listen for commands on stdin

--separator=SEPARATOR

Set separator character for menu values

--item-separator=SEPARATOR

Set separator character for menu items

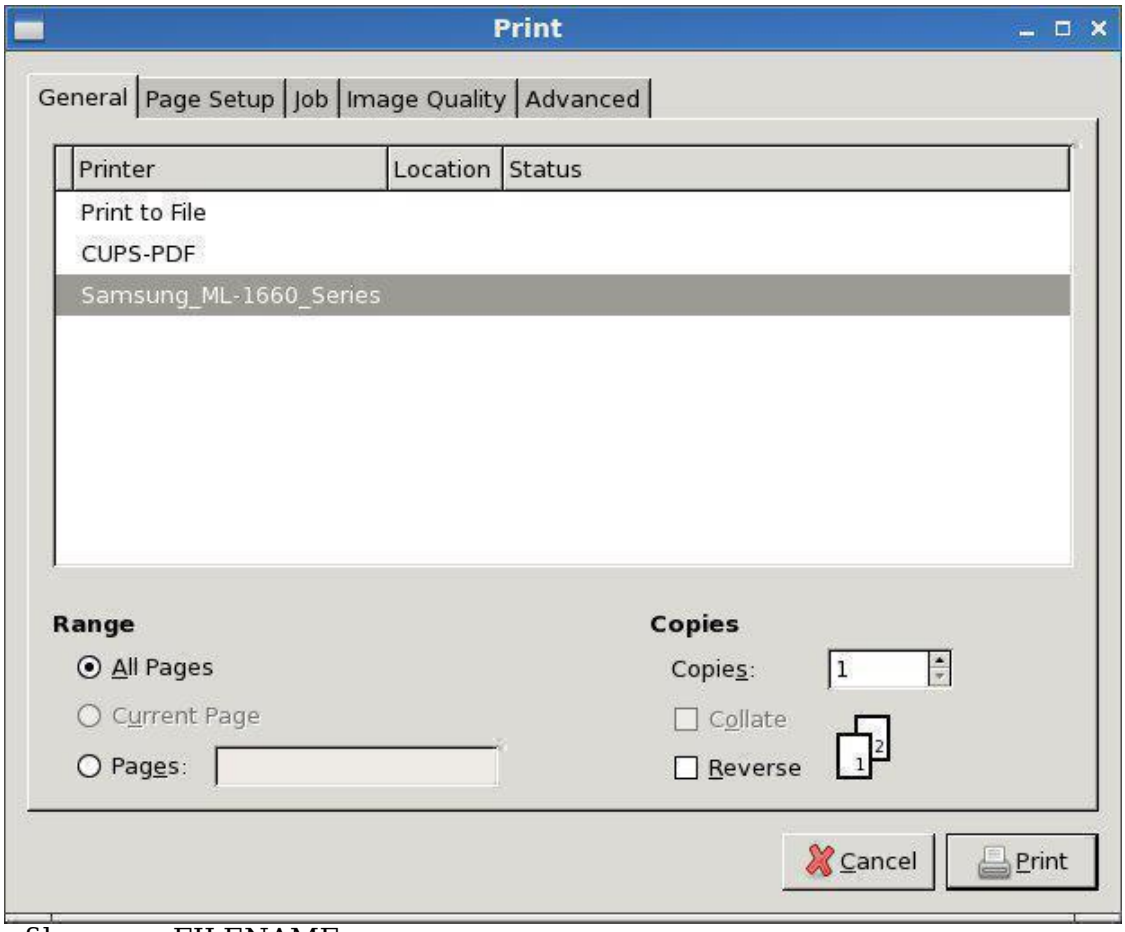
Print dialog options

**yad --print --filename=FILENAME**

To print a file called dogs.txt do:

**yad --print --filename=/root/dogs.txt**

This will invoke the print dialog ready to print the file dogs.txt.




--filename=FILENAME

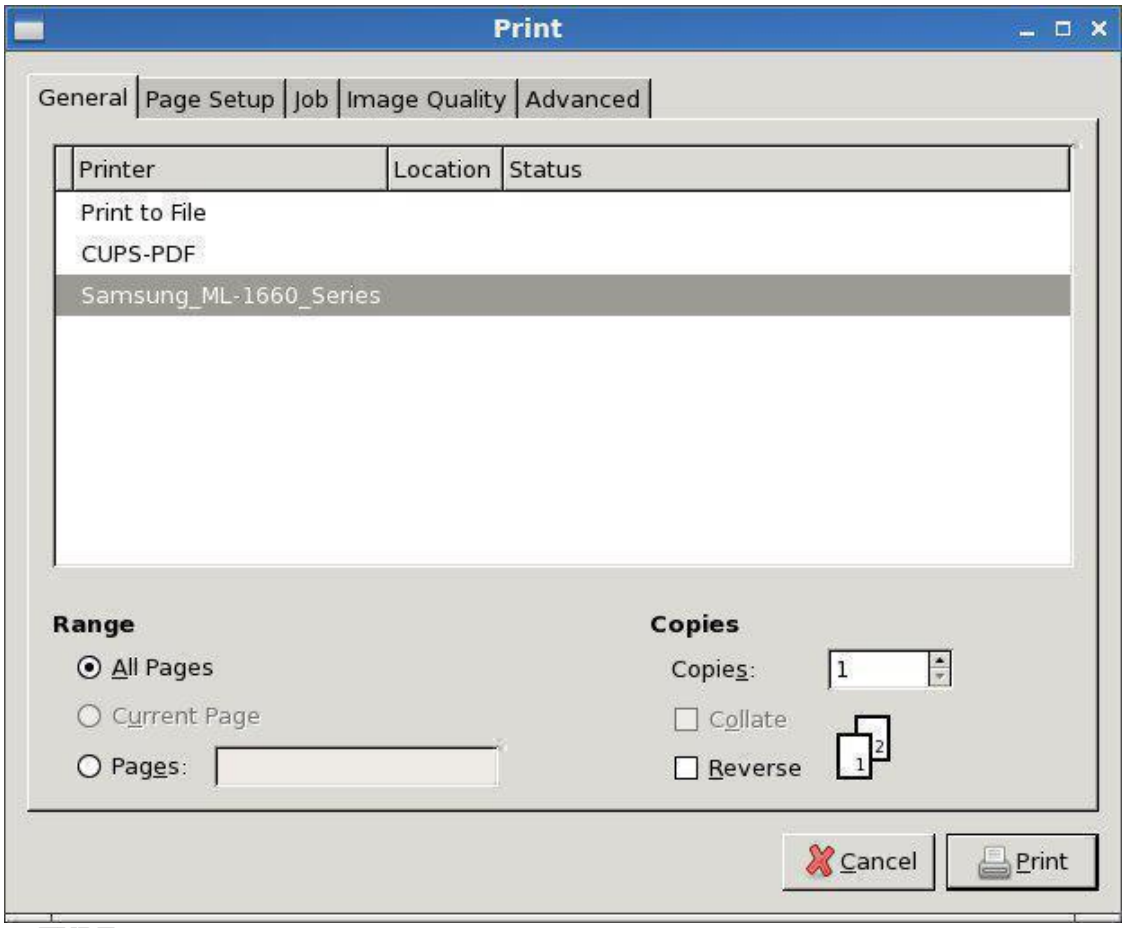
Source filename to print command.

The yad **--print** command does nothing alone.

You must add --filename=/PATH/FILENAME

**yad --print --filename=/root/dogs.txt**

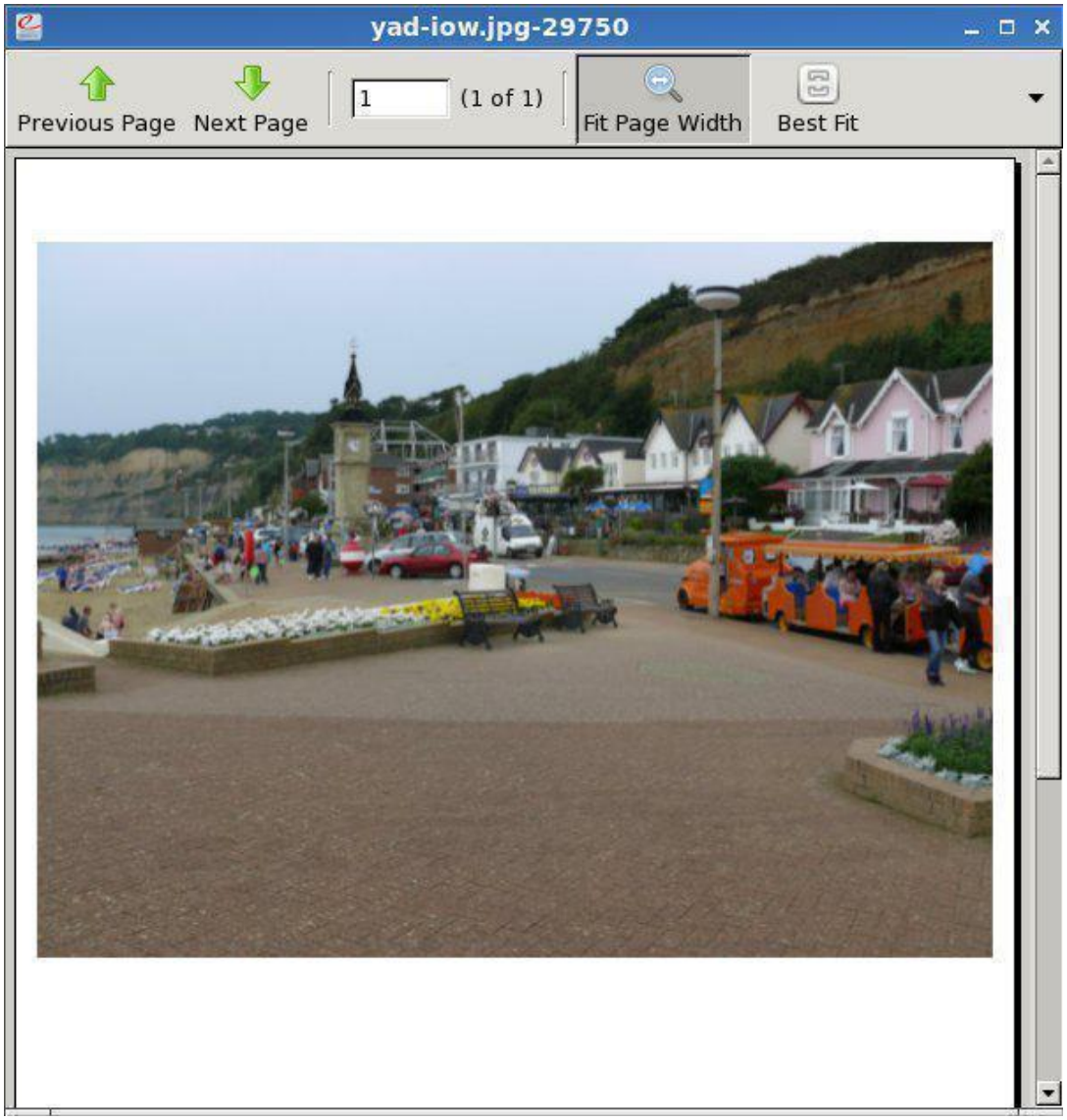
will load the file /root/dogs.txt into the print dialog.




--type=TYPE

Set source type (TYPE - TEXT, IMAGE or RAW)

**yad --print --filename=/root/iow.jpg --type=image --preview**




**--add-preview does not work with --type=raw but it will print.**

**yad --print --filename=/root/iow.pdf --type=raw**

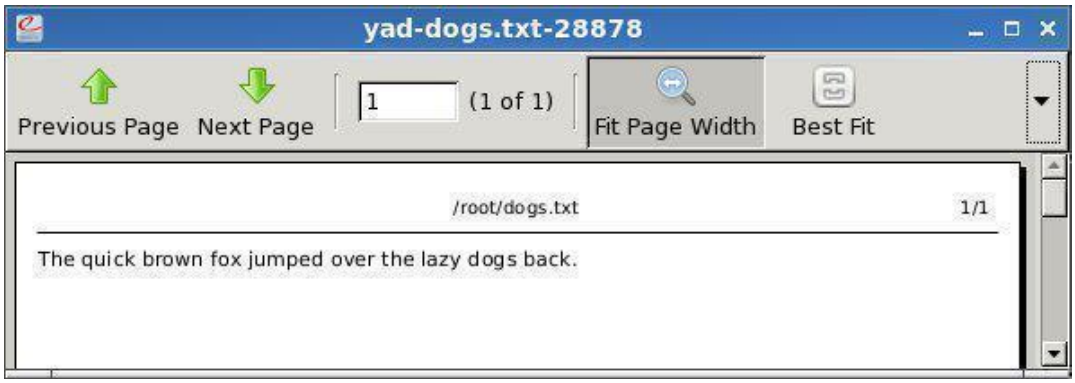
--headers

Add headers to pages.

The following command will add a Preview button to the print dialog, and add a header which will show the path/filename and page number.

**yad --print --add-preview --headers --filename=/root/dogs.txt**

This is what it looks like when viewed in evince when previewing.

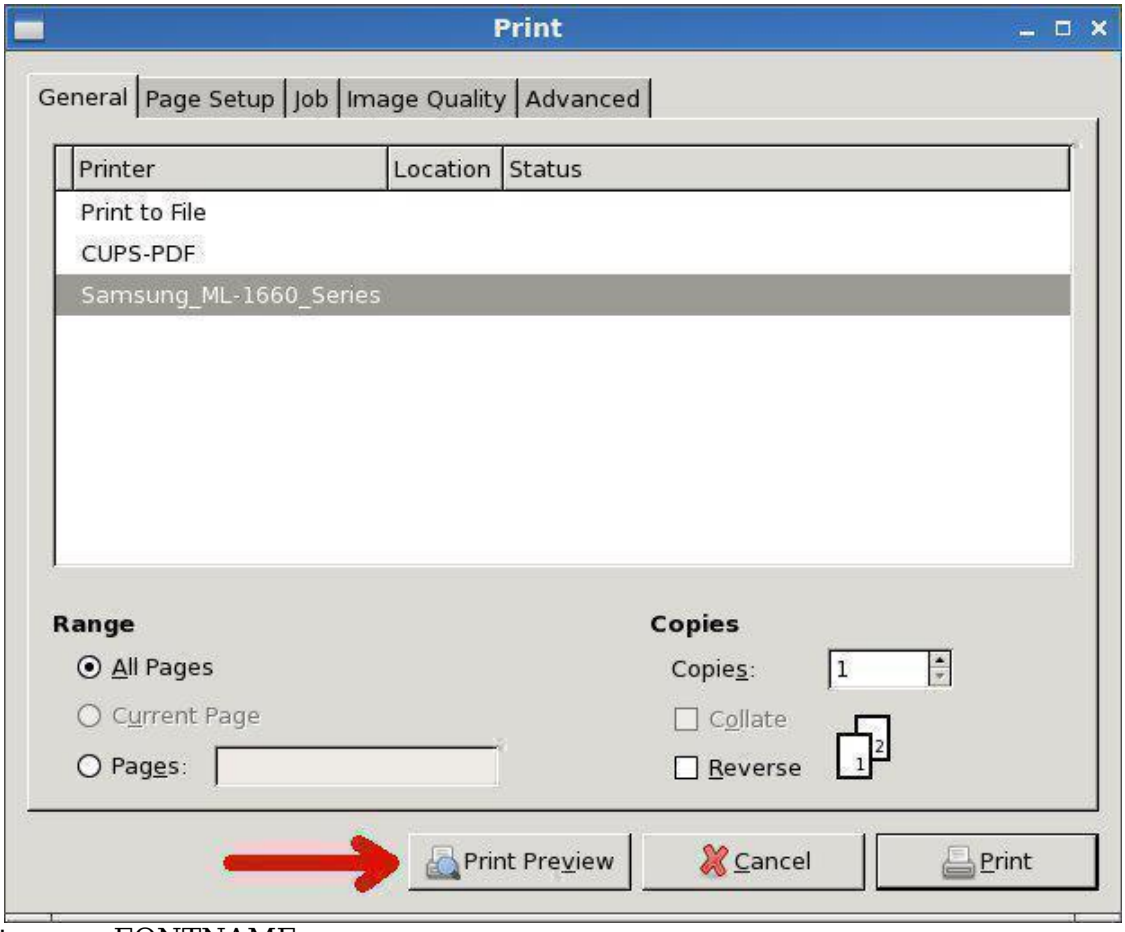



--add-preview

Enable preview in print dialog.

**yad --print --add-preview --filename=/root/dogs.txt**

Adds a Print Preview button to the print dialog.




--fontname=FONTNAME

Use specified font

Progress options

--progress-text=TEXT

Set progress text
--percentage=PERCENTAGE

Set initial percentage
--pulsate

Pulsate progress bar
--auto-close

Dismiss the dialog when 100% has been reached
--auto-kill

Kill parent process if cancel button is pressed
--rtl

Right-To-Left progress bar direction
Scale options
--value=VALUE

Set initial value
--min-value=VALUE

Set minimum value
--max-value=VALUE

Set maximum value
--step=VALUE

Set step size
--page=VALUE

Set paging size
--print-partial

Print partial values
--hide-value

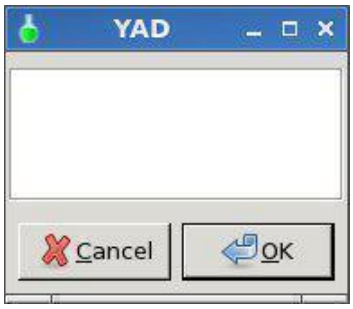
Hide value
--vertical

Show vertical scale
--invert

Invert direction
--mark=NAME:VALUE

Add mark to scale (may be used multiple times)
Text information options

**yad --text-info** is used for displaying information.




You will notice there is no =TEXT option.

**yad --text-info="Doesn't work"**

**yad --text-info < /root/my-documents/clipart/README-clipart.txt** does providing the file "README-clipart.txt" exists.

It reads the information from a file and displays it in the **--text-info box.**
--fore=COLOR

Use specified color for text.

You can change the text color by using the --fore command.

**yad --text-info --fore=red** but you have to generate text to display.

Lets create a simple text file called **hello.txt** in /root

Inside the text file type "Good morning"

Now read the file into the **--text-info** dialog like:

**yad --text-info --fore=red < hello.txt**


You can load quite a large text file and scroll bars will be added when required.
--back=COLOR

Use a specified color for background.

The background colour can also be changed like:

**yad --text-info --back=cyan < hello.txt**


Now lets try white text on a red background.

**yad --text-info --back=red --fore=white < hello.txt**


--fontname=FONTNAME

It's also possible to use a different font.

The font must be available, however.

**yad --text-info --back=red --fore=white < hello.txt**
**--fontname=Monospace**


Now for a few undocumented features.

Lets add italic, size and bold.

**yad --text-info --back=red --fore=white < hello.txt --fontname="**
**Monospace bold italic 20"**


Nice....

--wrap

Enable text wrapping.

Create a text file called dogs.txt. In the file type "The quick brown fox jumped over the lazy dogs back."

**yad --text-info --wrap < dogs.txt**

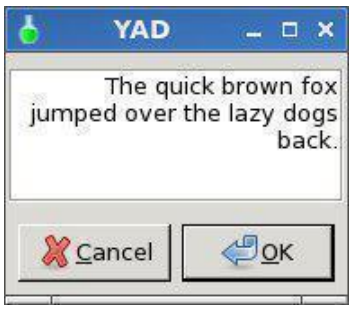

Notice how the text is wrapped in the dialog box.

--justify=TYPE

Set justification (TYPE = left, right, center or fill)

You can also justify the text. Let's try justifying the text to the right.

**yad --text-info --wrap --justify=right < dogs.txt**




Now center.

**yad --text-info --wrap --justify=center < dogs.txt**


Finally, fill.

**yad --text-info --wrap --justify=fill < dogs.txt**


--margins=SIZE

Set text margins.

**yad --text-info --wrap --margins=20 < dogs.txt**

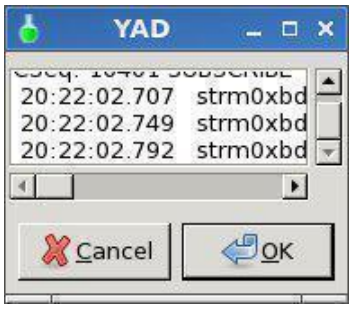

**I added --wrap so you could see margins on both sides.**

--tail

Autoscroll to end of text.

If you load a text file it will scroll through the text to the end of the file.

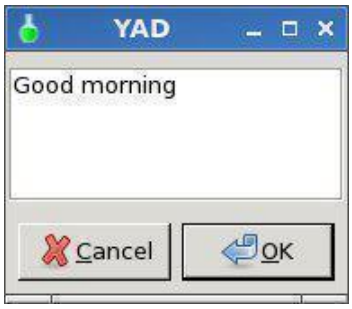
If it's a long file it will scroll too fast to read it.




It's a handy way to get to the end of a file.
--filename=FILENAME

Open a file in the **--text-info** dialog.

**yad --text-info --filename=/root/hello.txt**

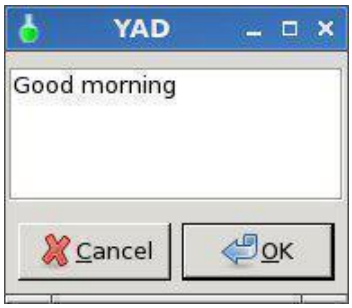



**yad --text-info < /root/hello.txt** will achieve the same result.

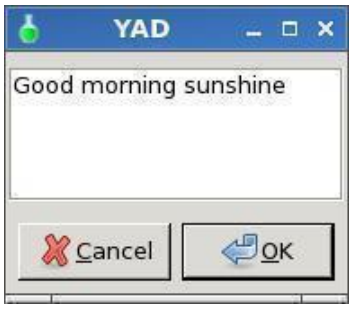
Good morning is the contents of the hello.text file.
--editable

Allow changes to the text.

**yad --text-info --editable < /root/hello.txt**




Now add sunshine after morning.




When you click OK, the following will be displayed:
**Good morning sunshine#**

Press the Enter key and you will be returned to the prompt.

Now lets make it do something a little more useful.

Write the following script, save it as testedit, then make it executable.

```
#!/bin/sh
# Created by smokey01
yad --text-info --editable < /root/hello.txt > /root/hello2.txt
rm /root/hello.txt
cp /root/hello2.txt /root/hello.txt
rm /root/hello2.txt
```

Now run **testedit** to have some fun.

If you don't want to edit the hello2.txt file just click OK.

If you click Cancel button the text in the file will be removed.
--show-uri

Make URI clickable

Create a file called links.txt in /root

Type **http://www.smokey01.com/menu** in the file and save the file.

**yad --text-info --width=250 < /root/links.txt --show-uri**


Now click on the link to go to the website.

Miscellaneous options

--about

**yad --about** displays the yad about dialog.


--version

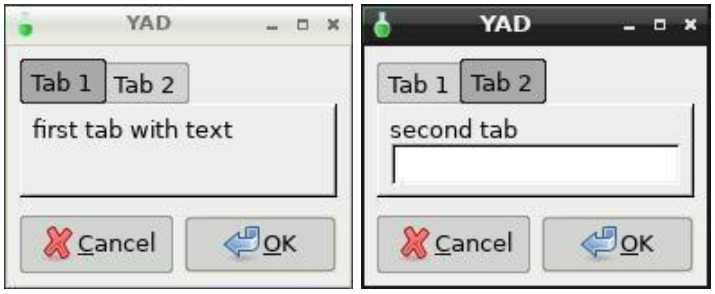
**yad --version** displays the yad version like:
0.36.1 (GTK+ 2.24.27)

Notebook

Notebook is a complex dialog which swallow other dialogs in his tabs. Dialogs identifies by unique key (integer) and must be runs in a special plug mode (--plug option). Following example runs notebook dialog with two tabs, first has a simple text and second is an entry dialog.

Copy and paste the following into a terminal.

**yad --plug=12345 --tabnum=1 --text="first tab with text" &> res1 &yad --plug=12345 --tabnum=2 --text="second tab" --entry &> res2 &yad --notebook --key=12345 --tab="Tab 1" --tab="Tab 2"**




GTK+ Options

--class=CLASS

Program class as used by the window manager

--name=NAME

```
            Program name as used by the window manager
      --screen=SCREEN

            X screen to use
      --sync

            Make X calls synchronous
      --gtk-module=MODULES

            Load additional GTK+ modules
      --g-fatal-warnings

            Make all warnings fatal
Application Options:
      --rest=FILENAME
            Load extra arguments from file
      --calendar
            Display calendar dialog
      --color
            Display color selection dialog
      --color-selection
            Alias for --color
      --dnd
            Display drag-n-drop box
      --entry
            Display text entry or combo-box dialog
      --file
            Display file selection dialog
      --file-selection
            Alias for --file
      --font
            Display font selection dialog
      --font-selection
            Alias for --font
      --form
            Display form dialog
      --icons
            Display icons box dialog
      --list
            Display list dialog
      --multi-progress
            Display multi progress bars dialog
      --notebook

            Display notebook dialog
      --notification
            Display notification
      --paned

            Display paned dialog
      --picture
```

Display picture dialog
--print
Display printing dialog
--progress
Display progress indication dialog
--scale
Display scale dialog
--text-info
Display text information dialog
--display=DISPLAY
X display to use

Functions

This is an example of a script with two functions in a form. The function command is not required but it can be used for clarity if you wish. Notice the difference between
example one and two.

```
#! /bin/bash
function on_click () {
yad --about
}
export -f on_click

function help () {
yad --text=" This is help file. "
}
export -f help

yad --title "Functions" --form --separator="" \
--button=gtk-ok:"bash -c on_click" \
--button=gtk-help:"bash -c help" \
--button=gtk-quit:0 \
```

The first function calls the About Dialog from the OK
button and the second function displays some help
when clicking on the Help button.

Functions can also be called without using the --form command.

```
#! /bin/bash
on_click () {
yad --about
}
export -f on_click

help () {
yad --text=" This is help file. "
}
export -f help
```

```
yad --title "Functions" --button=gtk-ok:"bash -c on_click" --button=gtk-
help:"bash -c help" --button=gtk-quit:0
```

To make the line a bit shorter on a simple GUI you can do this:

```
#! /bin/bash
on_click () {
yad --about
}
export -f on_click

help () {
yad --text=" This is help file. "
}
export -f help

yad --title "Functions" \
--button=gtk-ok:"bash -c on_click" \
--button=gtk-help:"bash -c help" \
--button=gtk-quit:0 \
```

Variables

Although YAD can produce output from it's many functions, it's a little more challenging
to export this infomation into variables than can be used with external applications.

The following script creates the GUI below.

```
#!/bin/sh
names=$(echo "Bill,George,Jack,Joe")
occupation=$(echo "Doctor,Dentist,Butcher,Baker,Candlestick Maker,Other")
yad --title="Names" \
--form --separator="," --item-separator="," \
--field="Names:":CB \
--field="Occupation:":CBE \
--field="Comments:":TXT \
"$names" "$occupation" "Type your comments here"
```


When you run the script and just press OK

you receive the following output when viewing
in a terminal:

**Bill,Doctor,Type your comments here,**

```
#!/bin/sh
names=$(echo "Bill,George,Jack,Joe")
occupation=$(echo "Doctor,Dentist,Butcher,Baker,Candlestick Maker,Other")
yad --title="Names" \
--form --separator="," --item-separator="," \
--field="Names:CB" \
--field="Occupation:CBE" \
--field="Comments:TXT" \
"$names" "$occupation" "Type your comments here" | while read line; do
NAME=`echo $line | awk -F',' '{print $1}'`
OCCUPATION=`echo $line | awk -F',' '{print $2}'`
COMMENT=`echo $line | awk -F',' '{print $3}'`

echo "(The selected name) (Selected occupation) (Comments)"
echo
echo
echo $NAME $OCCUPATION $COMMENT
done
```

Or

```
#!/bin/sh
names=$(echo "Bill,George,Jack,Joe")
occupation=$(echo "Doctor,Dentist,Butcher,Baker,Candlestick Maker,Other")
yad --title="Names" \
--form --separator="," --item-separator="," \
--field="Names:CB" \
--field="Occupation:CBE" \
--field="Comments:TXT" \
"$names" "$occupation" "Type your comments here" > /tmp/config

NAME=`cat /tmp/config | awk -F',' '{print $1}'`
OCCUPATION=`cat /tmp/config | awk -F',' '{print $2}'`
COMMENT=`cat /tmp/config | awk -F',' '{print $3}'`

#echo $NAME $OCCUPATION $COMMENT #this would have to be run first to work
properly with copy/paste into term, in a script it works ok

echo "(The selected name) (Selected occupation) (Comments)"
echo
echo
echo $NAME $OCCUPATION $COMMENT
```

Or

```
#!/bin/sh
names=$(echo "Bill,George,Jack,Joe")
```

```
occupation=$(echo "Doctor,Dentist,Butcher,Baker,Candlestick Maker,Other")
yad --title="Names" \
--form --separator="," --item-separator="," \
--field="Names:CB" \
--field="Occupation:CBE" \
--field="Comments:TXT" \
"$names" "$occupation" "Type your comments here" | while read line; do
echo "NAME='`echo $line | awk -F',' '{print $1}'`'" > /tmp/config
echo "OCCUPATION='`echo $line | awk -F',' '{print $2}'`'" >> /tmp/config
echo "COMMENT='`echo $line | awk -F',' '{print $3}'`'" >> /tmp/config
done
```

Examples

EXAMPLES
Display a file selector with the title Select a file to remove. The file selected is returned on standard output.

```
yad --title="Select a file to remove" --file-selection
```

Display a text entry dialog with the title Select Host and thetext Select the host you would like to flood-ping. The enteredtext is returned on standard output.

```
yad --title "Select Host" --entry --text "Select the host you would like to flood-ping"
```

Display a dialog, asking Microsoft Windows has been found! Wouldyou like to remove it?. The return code will be 0 (true in shell)if YES is selected, and 1 (false) if NO is selected.

```
yad --image "dialog-question" --title "Alert" --button=gtk-yes:0 --button=gtk-no:1
--text "Microsoft Windows has been found! Would you like to remove it?"
```

Show the search results in a list dialog with the title Search Resultsand the text Finding all header files....

```
find . -name '*.h' | yad --list --title "Search Results" --text "Finding all header files.."
--column "Files"
```

Show an icon in the notification area

```
yad --notification --image=update.png --text "System update necessary!" --command
"xterm -e apt-get upgrade"
```

Display a weekly shopping list in a check list dialog with Apples and Oranges pre selected

```
yad --list --checklist --column "Buy" --column "Item" TRUE Apples TRUE Oranges
FALSE Pears FALSE Toothpaste
```

Display a progress dialog while searching for all the postscript files in your home directory

```
find $HOME -name '*.ps' | yad --progress --pulsate
```

**A file filter to only select sfs files with multiple select.**
yad --title="Select a file to remove" --file-selection --file-filter="*.sfs"

Display a box with all of the installed desktop applications
yad --icons --read-dir=/usr/share/applications

yad --title="SFS Combiner v1" --text-align=center --text "Drag and drop your .sfs files here." --dnd | sed 's/^.......//' > /root/files

Use background and foreground colours, change font and font size.
**echo Ship | yad --text-info "Ship" --fontname="Monospace Italic 30" --fore=red --back=green --justify=left --wrap**

**yad --text-info --back=red --fore=white < hello.txt --fontname="Monospace bold italic 20"**

**lxsplit -j file.avi.001 &**
**while pkill -0 lxsplit; do**
**echo running**
**sleep 0.5**
**done | yad --progress --pulsate --auto-close --auto-kill --button gtk-cancel:1**

WD="/mnt/home/create-wd"
mkdir "$WD"

http://www.thelinuxrain.com/articles/multiple-item-data-entry-with-yad
Searching - look4

> This is a simple example of a GUI for the find command. It makes it easy to search for
> filenames using wildcards, between two dates and between two file sizes.
>
> #!/bin/sh
> #Written by smokey01
> #04 May 2016
>
> help () {
> yad --window-icon="gtk-find" --center --title="Look4 Help" --text="
> This is a simple little GUI made with
> YAD to help drive the find command.
>
> All fields must be populated or the GUI
> will not return any results.
>
> If you believe you should have received
> results but didn't, try widening the
> date and or size parameters.

Double click on results to open the file.

Requires YAD-0.36.2 or later.

Enjoy. smokey01
"
}
export -f help

look () {
loc=${1}
name=${2}
startdate=${3}
enddate=${4}
startsize=${5}
endsize=${6}
find $loc -name "$name" -newermt "$startdate" ! -newermt "$enddate" -size
+$startsize"k" ! -size +$endsize"k" | yad --width=600 --height=400
--separator=" " --window-icon="gtk-find" --title "Search Results" --center
--column "Files" --list --dclick-action="rox"
}
export -f look

yad --window-icon="gtk-find" --title="Look4 Files" --center --form --separator="
" --date-format="%Y-%m-%d" \
--field="Location:":MDIR "/root" \
--field="Filename:" "*" \
--field="Start Date:":DT "2000-01-01" \
--field="End Date:":DT "2016-12-31" \
--field="Start Size KB:":NUM "0" \
--field="End Size KB:":NUM "1024" \
--field="Find!gtk-find:BTN" 'bash -c "look %1 %2 %3 %4 %5 %6"' \
--button=gtk-help:'bash -c help' \
--button=gtk-quit:0