

# Tampil Indah dengan shell

Membuat aplikasi interaktif dan menarik di *console* tidak harus selalu menggunakan bahasa pemrograman yang rumit dengan memanggil pustaka yang rumit pula. Cobalah *shell script* terlebih dahulu...



**S**alah satu fenomena yang cukup menarik untuk di amati di dunia UNIX dan GNU/Linux adalah lengketnya pengguna dengan aplikasi console. Tidak peduli seberapa indah dan mudah Nautilus misalnya, pengguna kerap kali masih lebih memilih program ls misalnya. Berbagai alasan pun dikemukakan. Mulai lambatnya aplikasi berbasis GUI sampai kesetiaan dengan program-program di console. Walau, memang tidak setiap pengguna GNU/Linux lantas membenci keindahan yang ditawarkan oleh berbagai *desktop* yang indah-indah tersebut.

Sebagai seorang developer, tak jarang Anda menjumpai masalah-masalah yang seharusnya dapat diselesaikan dengan solusi yang tidak begitu rumit. Memberi peringatan kepada user-user di sistem ketika memasuki sesi sistem misalnya. Anda bisa saja meminta administrator sistem Anda untuk menambahkan sebaris perintah berikut:

```
echo "Halo $LOGNAME, jangan nakal-  
nakal  
ya..."
```

Akan tetapi, perintah tersebut yang mungkin dimasukkan ke dalam file *.bash\_profile* apabila Anda menggunakan BASH, rasa-rasanya kurang efektif walaupun dijalankan setiap kali user memasuki sesi sistem. Terlalu sederhana, terlalu tidak mencolok, terlalu kecil dan macam-macam alasan lainnya. User pun seringkali kelupaan membaca pesan-pesan satu baris semacam ini.

Lantas bagaimana? Membuat *newline* banyak-banyak di awal dan akhir pesan peringatan? Ini mungkin saja merupakan solusi. Akan tetapi, bagaimana dengan permasalahan lain

seperti ketika user-user Anda meminta *frontend* untuk program *cal*? Alasannya bisa saja karena program yang berguna untuk menampilkan kalender itu kurang informatif, kurang mudah dan macam-macam alasan lainnya.

Sekali lagi, lantas bagaimana? Mengabaikan saja permintaan user rasa-rasanya bukan hal yang cukup bijak, walau membuat aplikasi sendiri untuk mencukupi kebutuhan user mungkin lebih tidak bijak lagi, apalagi kalau Anda memiliki segudang kerjaan.

GNU/Linux hadir dengan sejumlah tool yang sebenarnya sangat berguna, akan tetapi terkadang sangat kurang dari sisi dandanannya. Tool-tool tersebut pun umumnya dapat menjadi solusi atas beberapa permasalahan. Jika ditambah dialog, atau *whiptail*, yang akan menjadi intisari dari artikel ini, berbagai tampilan indah akan muncul dan selamat tinggal hitam putih pun terucap. Semuanya menggunakan *shell script* dan tool-tool standar!

## Shell script

Kebanyakan distro saat ini datang bersama berbagai shell. Tinggal menyesuaikan selera, begitulah bahasa iklannya. Akan tetapi, apakah semua shell lantas mendukung shell script? Atau apakah shell script yang ditulis di BASH akan dapat terbaca di CSH misalnya? Atau apakah yang berhak menyandang gelar shell script hanyalah script yang dibuat oleh BASH, yang notabene adalah shell yang super komplrit?

Shell script hanyalah sekumpulan perintah (yang disimpan di dalam sebuah file) yang dijalankan oleh shell. Para veteran MS DOS akan menyebut ini kurang lebih seperti *batch file*. Pengguna MS DOS, yang mengenal satu jenis shell, yakni *command.com*

,tidak akan bertanya-tanya tentang portabilitas.

Sebuah shell yang akan menyandang predikat "dapat menjalankan shell script" haruslah komplan dengan standar POSIX. Oleh karena itulah, shell script yang Anda tulis di dalam shell yang menuruti standar POSIX tersebut, dapat dijalankan di shell manapun yang menuruti standar. BASH adalah contoh shell yang menuruti standar. Begitupun dengan shell-shell besar. Perbedaannya hanya pada perintah spesifik shell.

Terus bagaimana dengan shell-shell kecil? Penulis kerap kali menggunakan ASH, shell yang cukup kecil dan tetap mampu menjalankan shell script yang ada.

Berbagai shell default biasanya juga "mengaliaskan namanya" dengan *sh* untuk kompatibilitas.

## Dialog dan Whiptail

Ketikkan perintah berikut ini dan amati hasilnya:

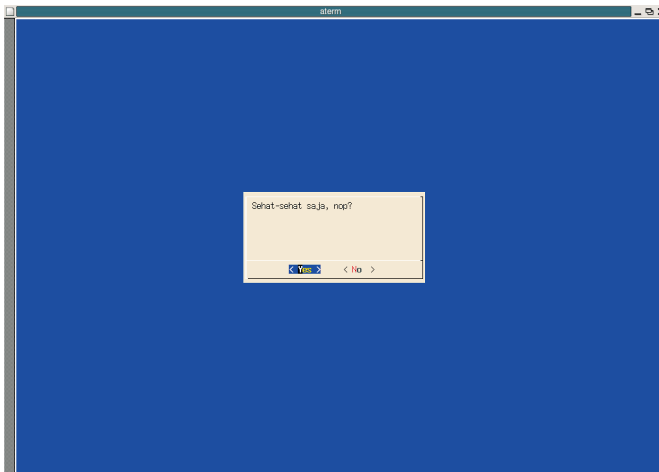
```
echo "Sehat-sehat saja, $LOGNAME ?"
```

Kemudian ketikkan perintah lain berikut dan amati pula hasilnya:

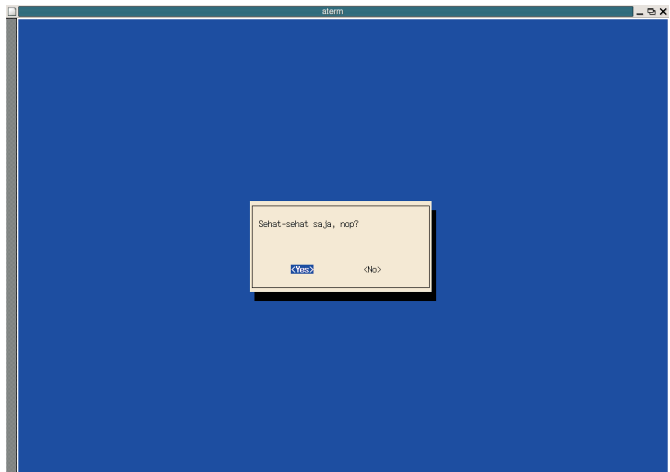
```
dialog --yesno "sehat-sehat saja,  
$LOGNAME?" 10 40
```

Apabila tidak terjadi pesan kesalahan, tentunya Anda bisa membandingkan tingkat keindahan dan interaktifitas di antara keduanya. Yang pertama hanya muncul dalam satu baris teks, sementara yang kedua hadir dalam sebuah kotak dialog lengkap dengan tombol YES dan NO. Keduanya ditulis dengan satu perintah!

Bagi Anda yang menemui pesan kesalahan semacam dialog: *command not found*, berarti program dialog belum terinstal pada sistem Anda.



🚩 Kotak dialog dengan dialog



🚩 Kotak dialog dengan whiptail

Kemudian gantilah kata dialog pada perintah yang kedua dengan whiptail dan cermati perbedaannya.

```
whiptail --yesno "sehat-sehat saja,
$LOGNAME?" 10 40
```

Keduanya memberikan tampilan yang kurang lebih mirip. Permasalahannya mungkin hanya pada masalah selera. Dialog dibangun di atas pustaka ncurses, sementara whiptail dibangun di atas pustaka newt.

Pada whiptail yang penulis pakai (0.50.17 - Debian 3.0r0), kekurangan masih terdapat di sana sini, terutama apabila dibandingkan dengan dialog. Untuk selanjutnya, di dalam artikel ini, kita akan menggunakan dialog. Bagi Anda yang jatuh cinta dengan whiptail, dalam beberapa contoh, Anda dapat mengganti tulisan dialog menjadi whiptail.

## Dialog 101

Dialog dibuat untuk memudahkan para pembuat shell script dalam meningkatkan keindahan dan interaktivitas aplikasi. Apa saja yang datang bersama dialog? Mari kita lihat satu per satu.

### • msgbox

Fitur yang satu ini mungkin termasuk dalam salah satu fitur standar. *Message box* dapat digunakan untuk memberikan pesan. Seorang administrator mungkin akan menyukai fitur

yang satu ini, terutama apabila digunakan untuk menarik perhatian user ketika memasuki sesi sistem.

Untuk memberikan pesan "halo" di atas sebuah kotak dialog berukuran 10x40 karakter, perintah berikut ini dapat digunakan.

```
dialog --msgbox halo 10 40
```

Secara *default*, sebuah tombol dengan label "OK" telah tersedia untuk Anda. Kurang suka? Lebih suka kata "yoi" untuk menggantikan "OK"? Cobalah perintah berikut ini:

```
dialog --ok-label yoi --msgbox halo 10 40
```

Masih kurang indah? Baiklah. Coba kita tambahkan judul di latar belakang. Sebuah opsi *--backtitle* dapat digunakan. Cobalah perintah berikut ini:

```
dialog --backtitle
"PESAN DARI
ADMIN" --ok-label
yoi --msgbox halo
10 40
```

Selain itu, beberapa opsi berikut ini mungkin menarik perhatian Anda:

➔ opsi *--title* *<str>* untuk menambahkan judul pada dialog (harap

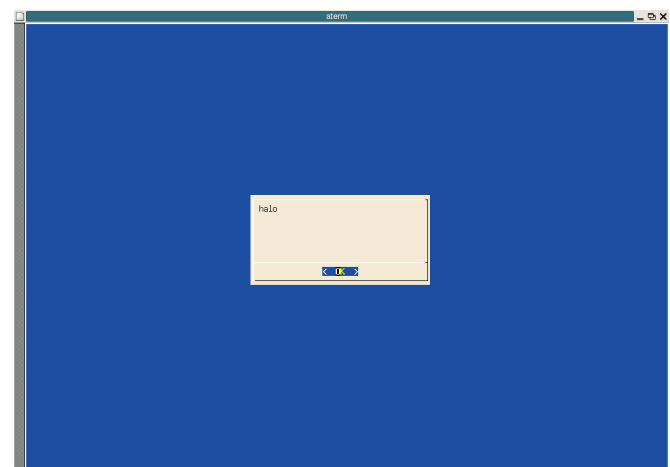
dibedakan dengan opsi *--backtitle*)

➔ opsi *--timeout <sec>* untuk memberikan batas waktu sebelum dialog Anda keluar secara otomatis. Sangat baik apabila digunakan untuk pesan yang tidak terlalu penting, yang akan menutup otomatis setelah beberapa lama diabaikan.

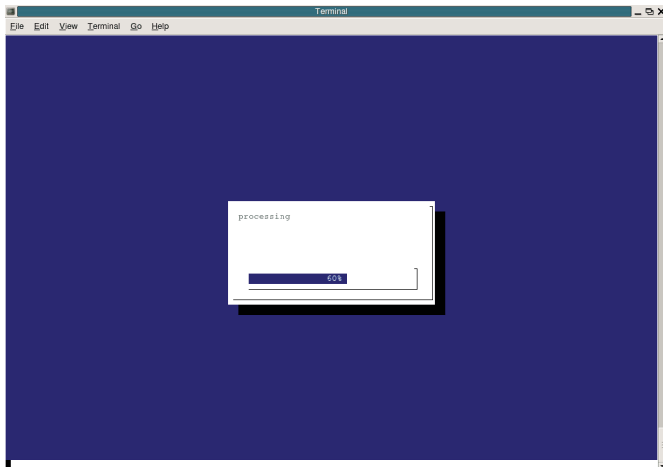
Sebagai catatan, opsi-opsi tersebut berlaku secara umum, termasuk untuk fitur lainnya.

### • textbox

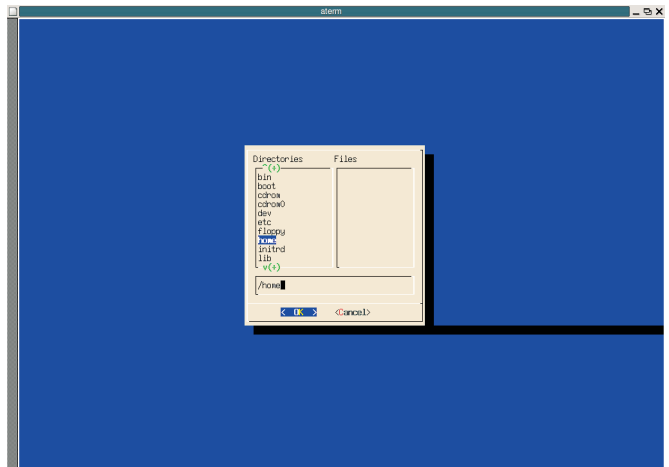
Fitur yang satu ini digunakan untuk menampilkan isi suatu file teks dalam dialog yang menarik. Anda mungkin bisa menampilkan pesan tentang lisensi atau hal-hal lain yang disimpan dalam file menggunakan fitur ini. Kurang terkesan? Tunggu dulu. Sekilas



🚩 Pesan dengan msgbox dialog



▲ Fitur gauge nan informatif



▲ Memilih file kini mudah dengan fselect

mungkin terkesan biasa-biasa saja. Akan tetapi, jika diamati lebih lanjut, fitur yang satu ini juga datang dengan persentase teks yang telah ditampilkan, dukungan scroll yang dapat diakses dengan tombol navigasi keyboard dan, *voila*: fitur *search* yang dapat diakses dengan menekan tombol / dan?

Terkesan? Cobalah fitur ini dengan memberikan perintah:

```
dialog --textbox /etc/passwd 10 40
```

## ● yesno

Kotak dialog *yes no* seringkali Anda temui di aplikasi berbasis GUI. Lain ceritanya dengan aplikasi console. Anda umumnya akan diharuskan menjawab yes/no dengan mengetikkan y atau n, bahkan yes atau no. Dengan fitur *yesno* yang disediakan, lupakan saja penekanan tombol y atau n tersebut.

Cobalah fitur ini dengan memberikan perintah:

```
dialog --yesno "Apakah Anda sehat-sehat  
saja, saudara $LOGNAME?" 10 40
```

Baiklah. Masalah mulai muncul. Bagaimana sang penulis shell script bisa mengetahui seorang memilih tombol Yes atau No? Untuk itu, program *dialog*, secara konsisten mengembalikan nilai 0 untuk penekanan OK atau Yes, 1 untuk Cancel atau No dan -1 untuk kasus-kasus lainnya.

Untuk memeriksa *return value*

perintah sebelumnya, Anda bisa mengevaluasi variabel *\$?*. Dengan demikian, blok perintah berikut ini akan melengkapi kotak dialog *yesno* kita menjadi lebih informatif.

```
dialog --yesno "Apakah Anda baik-baik  
saja, saudara $LOGNAME?" 10 40  
export retval=$?  
if [ $retval -eq 0 ]  
then  
    dialog --msgbox "Syukurlah  
$LOGNAME, Anda baik-baik saja" 10 40  
else  
    dialog --msgbox "Aduh ! Bagaimana  
ini $LOGNAME, apa yang Anda rasakan?"  
10 40  
fi
```

Bisa Anda cermati bahwa dengan fitur *yesno* dan *msgbox* kita bisa meningkatkan interaktivitas dan menjadikan aplikasi kita lebih informatif.

## ● inputbox dan passwordbox

Ingin meminta input dari *user*? Tidak masalah berkat datangnya fitur *inputbox*. Segala input akan disimpan ke dalam *stderr*. Berikan perintah berikut ini untuk mencoba:

```
dialog --inputbox halo 10 40
```

Fitur *inputbox* juga dilengkapi dengan kemampuan untuk memasukkan nilai default, yang dalam beberapa kondisi akan membantu user dalam memasukkan nilai tertentu. Fitur

*inputbox* ini serupa dengan *passwordbox* yang berguna untuk pengambilan informasi sensitif seperti halnya password. Perbedaannya adalah jika pada *inputbox*, teks yang dimasukkan akan dicetak di layar, maka pada *passwordbox*, teks yang dimasukkan tidak terlihat.

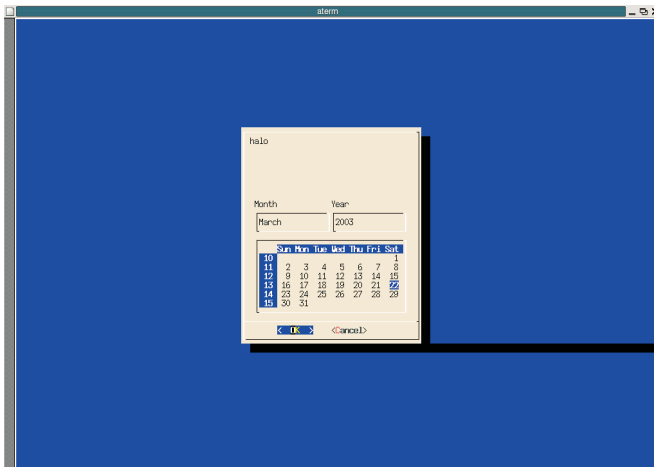
Jika pada *inputbox* pemberian nilai default akan sangat membantu user, maka pada *passwordbox* hal ini sangat tidak dianjurkan. Hal tersebut disebabkan karena pemberian nilai yang tidak terlihat oleh pemakai tentu saja tidak ada gunanya, di samping meningkatkan risiko melemahnya keamanan sistem karena pemberian nilai tersebut dapat diamati di dalam *process table*.

## ● gauge

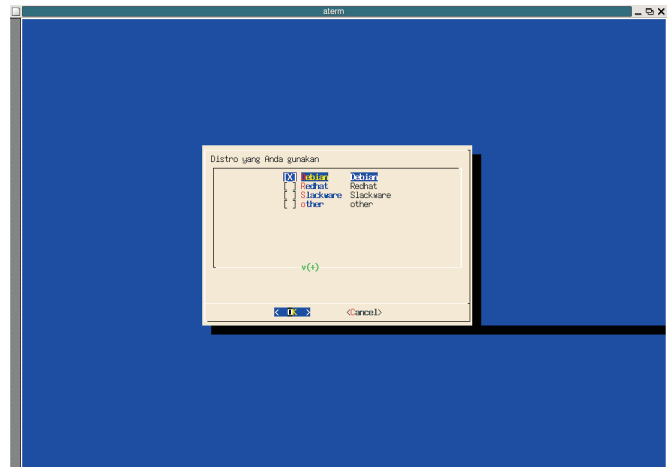
Memberikan informasi tentang seberapa jauh pekerjaan telah terselesaikan dapat Anda lakukan berkat adanya fitur *gauge* dari dialog. Gauge berfungsi seperti halnya sebuah *progress bar*. Nilai persentase dari gauge dapat dibaca dari *standard input*.

Berikut ini adalah blok perintah yang dapat Anda berikan untuk menguji fitur gauge:

```
i=0  
while [ $i -le 100 ]  
do  
    echo $i | dialog --gauge processing  
10 40  
    sleep 0.01
```



▲ Fitur calendar membuat shell script semakin indah



▲ Fitur checklist dari dialog

```
let i=$i+1
done
```

Dalam contoh tersebut, kita menggunakan perintah `sleep` yang berfungsi untuk menunda suatu anak proses (di mana dalam contoh tersebut waktu tunda adalah 0.01 detik). Nilai variabel `i` tersebut kita cetak dan berikan kepada program `dialog` yang menerimanya sebagai standard input untuk `gauge`.

### ● fselect

Ini dia fitur yang luar biasa dari `dialog`. Sebuah file selector! Dengan fitur ini, Anda bisa memanjakan user Anda dengan mengizinkannya untuk memilih nama file dengan antarmuka yang indah daripada meminta user Anda untuk mengetikkan serangkaian path yang panjang. Sama seperti halnya `inputbox`, keluaran dari `fselect` akan disimpan di dalam `stderr`.

Berikut ini adalah perintah untuk mencoba `fselect`:

```
dialog --fselect / 10 40
```

### ● calendar

Seperti kasus yang telah dibahas bahwa program `cal`, walau telah memiliki tampilan yang bagus, tetap kurang interaktif, maka dengan fitur `calendar` dari `dialog`, semua masalah telah terpecahkan. Fitur yang satu ini mengizinkan Anda menampilkan kalender saat ini ataupun masa depan

dan masa lalu serta mengembalikan nilai dalam format tanggal/bulan/tahun ketika user keluar dari kotak dialog ini.

Berikan perintah berikut ini untuk mencoba `calendar`:

```
dialog --calendar halo 10 40
```

### ● radiolist

Setelah memanjakan kita dengan berbagai fitur yang telah dibahas, kini `radiolist` datang untuk menambah serangkaian fitur. Sesuai dengan namanya, `radiolist` menyediakan kemampuan untuk membentuk serangkaian pilihan dimana hanya memperbolehkan satu pilihan aktif. Sama seperti fitur lainnya, nilai terpilih dicetak di `stderr` ketika user mengakhiri dialog.

Ketikkan contoh berikut ini untuk mencoba `radiolist`:

```
dialog --radiolist "Distro yang Anda gunakan" 20 60 10 Debian Debian on Redhat Redhat off Slackware Slackware off other other off
```

### ● checklist

Apabila `radiolist` hanya mengizinkan satu pilihan aktif, maka `checklist` mengizinkan pemilihan beberapa item sekaligus. Ketika user mengakhiri dialog, semua pilihan akan dikembalikan dalam format "pil1" "pil2" "pil3" .... Nilai kembalian tersebut dalam diproses di dalam iterasi `for` ataupun dapat dipisahkan dengan program `cut`.

Ketikkan contoh berikut ini untuk mencoba `checklist`:


```
dialog --checklist "Distro yang Anda gunakan" 20 60 10 Debian Debian on Redhat Redhat off Slackware Slackware off other other off
```

### ● menu

Katakanlah user harus memilih satu dari beberapa pilihan. Dalam situasi seperti itu, Anda bisa menggunakan `radiolist`, walau kurang memenuhi antarmuka yang ramah untuk pemakai. Untunglah `dialog` juga menyediakan fitur menu yang dapat digunakan untuk pembuatan menu. Nilai kembalian dicetak ke dalam `stderr`.

Ketikkan contoh berikut untuk mencoba menu:

```
dialog --menu "Kegiatan sehari-hari" 20 60 10 makan makan minum minum tidur tidur
```

Luar biasa! Hanya itulah kesan yang terucap untuk program yang satu ini. Dengan `dialog`, beberapa solusi yang memerlukan antarmuka ramah dapat dibuat hanya dalam hitungan menit. Bayangkan apabila Anda harus membuatnya dengan berbagai bahasa pemrograman dilengkapi pula dengan penggunaan pustaka `ncurses`. Waktu yang dihemat bisa Anda gunakan untuk keperluan lainnya. Bagi Anda yang bermain di GNOME, tersedia pula `gdialog` yang menawarkan antarmuka lebih ramah lagi. Salam. 

**Noprianto** ([noprianto@infolinux.co.id](mailto:noprianto@infolinux.co.id))