Olimpiada de inovare și creativitate digitală - InfoEducație

an școlar 2021-2022

pearOS NiceC0re

Autor:

- **BĂLAN ALEXANDRU**
- **clasa a XII-a**

Profesor coordonator:

- **Merlan Doina Narcisa**

Colegiul Economic, Rm. Vâlcea

DOCUMENTAȚIE

Ce reprezintă lucrarea mea?

Lucrarea mea reprezintă un Sistem de Operare, mai exact o distribuție Linux, bazată pe Arch, care se folosește de Kernel-ul Linux și Bootloader-ul GRUB2.

Cum se cheamă distribuția mea linux?

pearOS

De ce acest nume?

Mi s-a părut amuzant ca în loc de un măr mușcat să folosesc o pară mușcată, având în vedere că distribuția mea se aseamănă într-un fel sau altul cu macOS de la Apple.

Scurt istoric:

pearOS a început acum multi ani, dezvoltat de francezul David Tavares, acesta abandonand proiectul în Martie 2016. Proiectul a avut multe probleme referitoare la licenta GPL cu care a fost lansat.

David Tavares folosea o aplicație care face backup-uri bootabile pentru a creea distribuția de Linux, pearOS, care, la acea vreme era Ubuntu cu teme de macOS si nimi mai mult, codul sursa nefiind disponibil pentru public(ceea ce incalca licenta GNU Public License).

Dupa ce David Tavares a abandonat proiectul, in 2019 eu am devenit noul owner al pearOS. David Tavares este acum in proiectul Trenta OS, iar eu dețin toate drepturile de autor(copyright) asupra proiectului.

Ce am luat, mai exact, din vechiul pearOS? Doar numele a fost preluat de la vechea distribuție. Ideea vechiului proiect a fost o distribuție Linux care sa semene cu iOS(sistemul de operare mobil de la Apple). Ideea proiectului curent este aceea de a oferi cea mai smooth experiența Linux care sa se asemene cu UI-ul si UX-ul sistemului de operare de la Apple, și anume macOS.

Ce aduce pearOS în plus față de alte distribuții linux?

În general, când vorbim despre Linux, ne gândim la o simplă interfață în linia de comandă, sau un UI destul de învechit, oricum ar fi, în general, pe linux este foarte folosit terminalul. Ceea ce dorește pearOS să aducă, este o interfață foarte fluentă și ușor de utilizat, fără a mai avea nevoia aceasta masivă de accesul la un terminal / o linie de comandă, totul fiind automatizat sau simplificat prin diverse aplicații și scripturi, cum este scriptul de generare a imaginii ISO a sistemului de operare, în loc de o întreagă suită de comenzi, am ales un script cu un meniu interactiv, foarte ușor de înțeles și folosit, chiar și de către începători.

Ce planuri de viitor am, relative cu pearOS?

- Să fie cea mai cunoscută distribuție Linux.
- Să fie cat mai ușor de folosit, fără a fi nevoie să cunoaștem termeni “complicati” de informatică.

• Să fac cat mai multe componente “home-baked” (făcute de mine), cum ar fi un pachet de Office.

Ce am ales pentru a crea pearOS, si de ce?

Am ales Arch Linux, deoarece este o bază foarte goală (are nevoie de componente instalate manual), este o bază rapidă (având în vedere că integrez exact ce pachete doresc), foarte important pentru User Experience: sistemul de versioning este acela Rolling Release, nefiind nevoie de reinstalarea OS-ului la fiecare actualizare majoră.

Plus de asta, Arch Linux vine și cu un script ajutător care generează imagini ISO (archiso), fără acesta, creearea oricărei distribuții linux fiind imposibilă.

Cel mai important, pearOS se bucură de tehnologii și componente Open Source, nicio componentă listată în pearOS nu are sursa închisă (proprietary, closed source software).

Ca notă de final de capitol, pearOS are și propria licență de Software Open Source, sub care majoritatea aplicațiilor din pearOS sunt lansate, și anume, Pear Public License V2.

Caz particular: Laptop-urile de la Clevo

pearOS are și suport pentru laptop urile de la Clevo, prin aplicația componentă Cyrex, o aplicație care setează RGB-ul tastaturii și controlează cooler-ul de pe procesor.

Conectivitatea pearOS-ului:

Mulțumită bazei foarte versatile și a software-ului open source, pearOS are suport pentru TOATE plăcile de rețea, cele mai potrivite fiind cele de la Realtek, Intel și Broadcom, precum și plăcile de Bluetooth.

pearOS mai poate fi conectat la internet și prin cablu de rețea (ethernet), beneficiind de viteza maximă permisă de placa de retea și de internet provider.

Unde poate fi pearOS găsit pentru a fi descarcat si instalat?

pearOS poate fi găsit pe site-ul https://pearos.xyz, un site creat de mine împreună cu un baiat din Cluj, pe nume Andrei Muntean.

Acest sistem de operare, fiind Community-Oriented, primește suport de la comunitate. Asta însemnând că oamenii pot face site-ul și aplicații pentru sistemul meu de operare fără nicio obligație. Pe scurt, Andrei Muntean a vrut să facă o faptă bună și să ajute proiectul oferind ajutor în crearea site-ului, el ocupându-se de structura site-ului, animații și mici detalii referitoare la alegerile de culori.

Ce oferă distribuția mea Linux?

Distribuția mea oferă un mediu de utilizare al computer-ului diferit și în același timp foarte familiară, asemănându-se cu macOS de la Apple.

Este ușor de folosit, distribuția având o interfață grafică (GUI) care este KDE (K Desktop Environment) si KWM (K Window Manager), dar având și o linie de comandă, aceasta putând

fi accesată printr-un terminal (Similar cu CMD pe Windows) sau o consola de tip TTY (similară cu ce avea Microsoft să aducă înainte de Windows, și anume, DOS: o consolă pe tot ecranul).

Cum se instaleaza pearOS?

- Intră pe site-ul https://pearos.xyz
- Deschide secțiunea `Versions`
- Alege `NiceC0re`
- Imaginea ISO va fi descărcată
- Descarcă o unealtă care face un stick USB bootabil (eu prefer Balena Etcher)
- După ce s-a creeat USB bootabil, reporniti calculatorul in Boot Menu
- Bootați de pe USB
- Folositi aplicatia de instalare a pearOS
- Acceptați termenii și condițiile, selectați disk-ul unde pearOS va fi instalat.
- După instalare, un “Expert de Configurare” o să vă solicite crearea unui utilizator local și o parolă
- Reporniți computer-ul și sunteți gata pentru a folosi pearOS.

Stabilitate, performanțe, securitate

Având în vedere că pearOS este o distribuție serioasă, cu planuri reale de viitor, dezvoltată și menținută de o singură persoană (singurul ajutor fiind cel de la comunitate), el mai conține și bug-uri, ceea ce e normal, mai ales că nu sunt bug-uri grave (de securitate, integritatea computer-ului)

Deoarece este o distribuție Linux, performanțele în utlizare cresc foarte mult, mulțumită driverelor de tip Open Source și suportul mult mai rapid.

Securitate, vulnerabilitati:

pearOS se folosește întotdeauna de cele mai noi versiuni de kernel și de aplicații, făcând munca hackerilor mult mai grea, aceștia găsind mai greu exploit-uri.

În caz ca apare vreo eroare neprevăzută, log-urile și mesajele de eroare sunt mereu ușor de găsit/văzut/citit.

Versiuni:

NiceC0re vine ca o versiune a pearOS (exemplu Windows XP, Windows 7 etc; pearOS ThiccSur, pearOS Monterey, pearOS NiceC0re).

Iar aceste versiuni au la rândul lor sub versiuni (de exemplu Ultimate, Starter, Home de la Windows 7), valabile pe GitHub (https://github.com/pearOS-archlinux/iso).

Care este range-ul de varsta mediu?

pearOS este folosit într-o plajă foarte mare de varste, cei mai tineri utilizatori fiind copii pasionați de 11-13 ani, cei mai în vârstă fiind cunoscători și experimentați, de pana la 60 de ani.

Este pearOS fluid și rapid?

Am pus foarte mare accent pe aspect, fluiditate, responsivitate, viteza și adaptivitate. Un răspuns scurt este DA, pearOS suporta toate rezoluțiile de display moderne, animatii și UI plăcut vizual (fără contraste greșite, greu vizibile, neintuitive).

Ce oameni targetează pearOS?

În opinia mea, pearOS este pentru oricine, indiferent de vârstă sau experiență. Chiar și un neexperimentat se poate descurca să instaleze + folosească pearOS. În cazurile în care cineva are o problemă cu ceva legat de pearOS (componentă, aplicație etc.), există suport pe toate rețelele sociale ale pearOS, și anume YouTube, Instagram, WhatsApp, email, Twitter.

Unde raportez bug-urile descoperite?

Pe GitHub, la tab-ul “ISSUES”, sau pe server-ul meu de MatisBT disponibil pe https://bugs.pearos.xyz

Ce am nevoie pentru a rula pearOS pe computer-ul meu?

- un procesor cel puțin Dual Core
- ce puțin 4GB ram
- un HDD de cel puțin 20GB
- un stick USB (pentru mediul bootabil)
- o conexiune stabilă la internet

Ce componente utilizează pearOS? (design și aplicații)

În primul rand am folosit X11 ca și Window Server. În prezent există 2 mari servere de display: Wayland și X11.

Wayland este mai nou, mai rapid, mai fluent, dar dezavantajul este stabilitatea, fiind motivul pentru care am optat pentru X11, un Display Server mai vechi, dar care își face treaba fără probleme.

Peste Serverul de display, în general, avem nevoie de un Desktop Environment cu un Window manager, sau un Window Manager independent. Alegerea mea a fost una simpla: K Desktop Environment, deoarece pot dezvolta widget-uri pentru acesta, poate fi modificat în aproximativ orice fel (eu am ales un design cu foarte mult Blur și Transparenta), iar ca Window Manager, am folosit varianta prestabilita, și anume KWin (K Window Manager), deoarece se integrează perfect cu KDE, având posibilitatea de a implementa animatii și efecte pentru ferestre.

Pentru design, m-am folosit de tema “Sweet Mars” (https://store.kde.org/p/1393507) pe care am integrat-o în pearOS cu mici modificări.

Pentru iconițe, pearOS se bucură de pachetul de iconițe Bubble Dark, modificat de mine (pachetul original poate if gasit aici https://github.com/yeyushengfan258/Bubble-icon-theme). În baza acestui Icon Pack, am creat un pachet de iconițe cu tema întunecată.

Pentru Dock (bara de jos, cunoscută ca și TaskBar) am folosit Latte Dock, cu o configurație personalizată făcută de la 0, la fel și pentru bara de sus (bara de meniu).

➢ Instalatorul

Instalatorul (expertul de instalare) este împărțit în două:

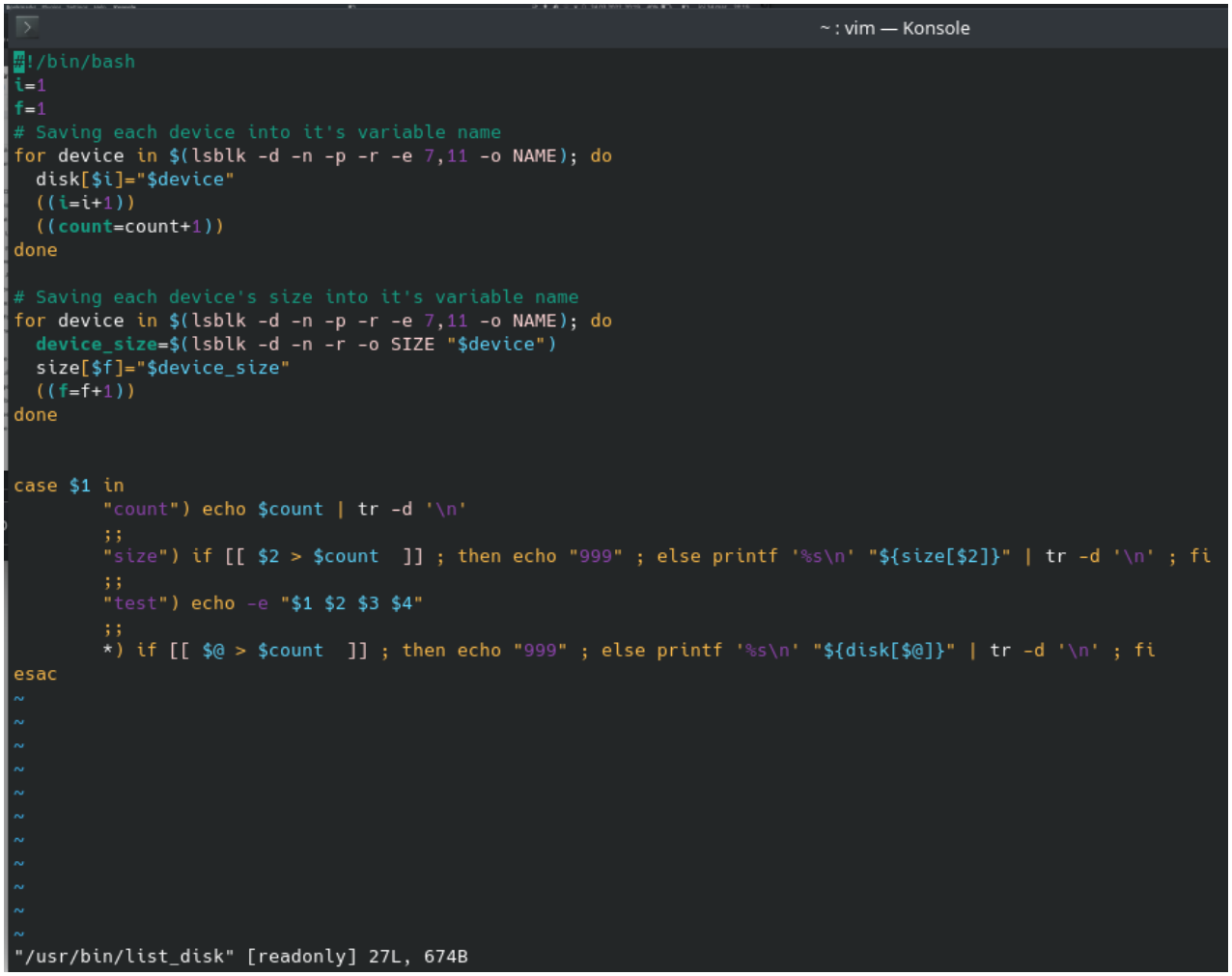
- Partea care formateaza disk-ul, creeaza sistemul de fișiere și instalează aplicațiile necesare.

- Partea care crează utilizatorul local (local computer account), care cere numele de utilizator și o parola.

După ce instalatorul primește numele complet, utilizatorul și parola, acesta finalizează diverse setări pentru a optimiza sistemul de operare.

Pentru aplicația de instalare, m-am gandit sa folosesc Node JS și bash, deoarece pot folosi HTML și CSS pentru a efectua interfața aplicației. Pentru design, am plecat de la o structură în HTML, găsită pe internet.

Pentru a lista disk-urile, am creeat un script bash, care listeaza diverse informatii:



Acest ^ bash sript se afla in `/usr/bin/list_disk` si este folosita in JavaScript astfel:

```
////////////////////////////////////////////////////////////////////////////////////////////////////

function list_disk() {
//vars:
const { exec } = require('child_process');
count = 0;
i = 1 ;
f = 0;
var z=``;
var diskname = "";
var disksize ="";

exec('list_disk count', (err, numberofdisks) => {
var sCount = `${numberofdisks}`
count = parseInt(sCount);
console.log("Available disks are:" + count);

while (i < (count+1)) {
        console.log("THE VALUE IS " + i);
                exec("list_disk " +i, (err, stdout) => {
                var f=1;
                var zi=`
                <li>
                  <label class="label_for_disk">
                    <input type="radio" id="disk` +(i-count) + `" name="disk" value="${stdout}">
                    <img class="disk_logo" height=50px src="../../resources/disk.png"></img>
                    <p id="label_disk` +(i-count) + `" class="disk_title">${stdout}</p>
                  </label>
                </li>
        `;
        i++;
        z += zi;
        document.getElementById("disk_list").innerHTML =z;
        })
        i++;
        }
        })
}
```

Mai exact, am importat `child_process` sub constanta `exec` din JavaScript, si am solicitat outpu-ul script-ului bash, care este salvat in `numberofdisks` pentru a numerota disk-urile, numarul disk-urilor fiind salvat in variabila `count`.

Pentru a lista disk-urile, am definit `i = 0`, si apoi un loop are printeaza numele disk-urilor de `i` ori (`i` fiind `count + 1`, `i` crescand cu fiecare loop):

```
while (i < (count+1)) {
        console.log("THE VALUE IS " + i);
                exec("list_disk " +i, (err, stdout) => {
                var f=1;
                var zi=`
                <li>
                  <label class="label_for_disk">
                    <input type="radio" id="disk` +(i-count) + `" name="disk" value="${stdout}">
                    <img class="disk_logo" height=50px src="../../resources/disk.png"></img>
                    <p id="label_disk` +(i-count) + `" class="disk_title">${stdout}</p>
                  </label>
                </li>
        `;
        i++;
        z += zi;
        document.getElementById("disk_list").innerHTML =z;
        })
        i++;
        }
        })
}
```

Pentru a selecta DISK-ul m-am folosit de metoda Radio Buttons, insemnand ca un user nu poate selecta mai mult de un disk(problema care ar fi dus la imposibilitatea de a instala si implicit rula pearOS). Codul care se ocupa cu selectia si salvarea disk-urilor este urmatorul:

```
function select_disk() {
        var radios = document.getElementsByName('disk');
        for (var i = 0, length = radios.length; i < length; i++) {
          if (radios[i].checked) {
          const fs = require('fs');
          fs.writeFileSync('/tmp/disk-to-install', '' + radios[i].value);
          // starting the shell //
          const { exec } = require('child_process');
          exec("sudo setup " + radios[i].value + "&> ~/Desktop/install.log", (err, stdout) => {
          })
        // ending the shell //
        break;
        }
    }

}

var p = document.getElementsByTagName("p");
```

Mai exact, salveaza disk-ul selectat (valoarea radio-button) intr-un temp file(fisier temporar), apoi se solicita un `exec care` ruleaza scriptul bash de instalare a distributiei.

Scriptul bash de instalare este simplu de inteles:

- colectarea datelor de sistem in variabile:

```
# Setting usefull variables
DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
BASE_PACKAGES=('alsa-utils' 'amd-ucode' 'arch-install-scripts' 'b43
UEFI=false
bluetooth=false
installed=false
mounted=false
architecture=$(lscpu | grep 'Architecture' | awk '{print $2}')
hypervisor=$(systemd-detect-virt)
```

- Setarile utilizatorului temporar, in variabile:

```
# Various settings for the installation process
KEYMAP="us"
LOCALE="en_US.UTF-8"
utc_enabled=false
ZONE="Europe/Bucharest"
HOST_NAME="pearOS-live"
user_passwd="pear"
root_passwd="pear"
FULL_NAME="Default User"
USER_NAME="default"
SWAP="-"
swap_enabled=false
block_devices=()
PARTITION_LAYOUT="Basic"
FILE_SYSTEM="ext4"
DISK="$1"
PREFIX=""
multilib=true
DESKTOP="None"
DESKTOP_PACKAGES+=("${DESKTOP_DEFAULTS[@]}")
DESKTOP_PACKAGES+=('plasma-nm')
```

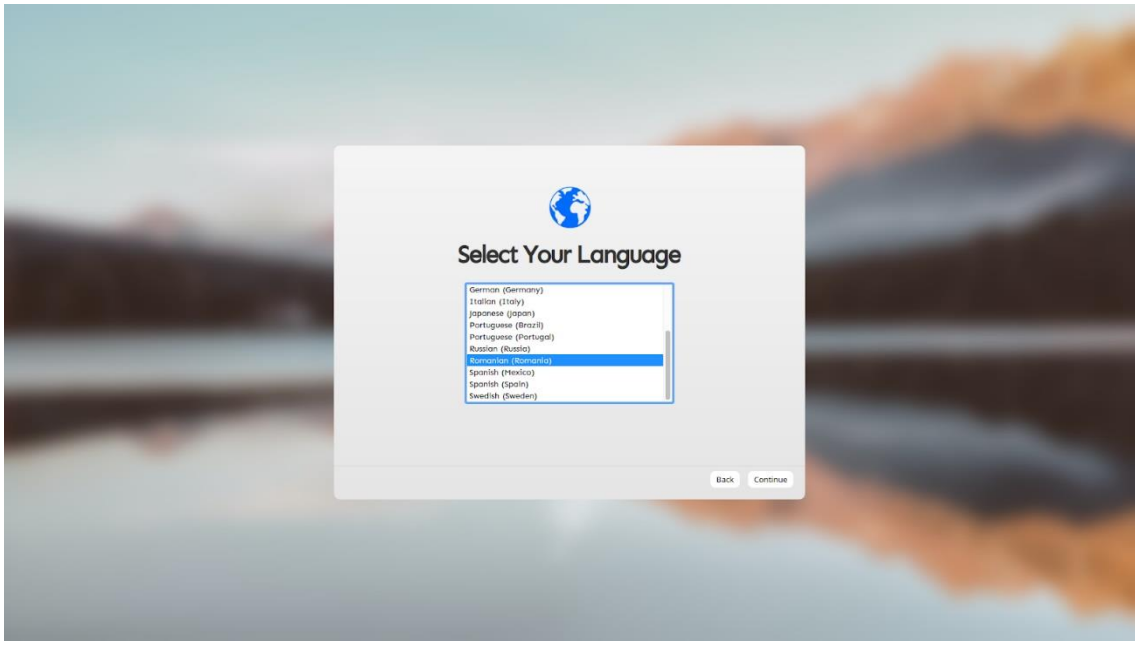
Diverse verificari de sistem (daca are bluetooth, daca este UEFI sau Legacy/BIOS), daca sistemul foloseste SSD sau HDD (linux noteaza diferit SSD-urile conectate cu NVMe, acesetea adaugand un mic prefix la path-ul de disk (de exemplu `/dev/nmve0n1p`):

```
# Check if the system is UEFI or legacy
if [ -d /sys/firmware/efi ]; then
  UEFI=true
fi

# Check if the system has bluetooth or not
if dmesg | grep -iq "blue"; then
  bluetooth=true
fi

# Simple fix for the NVME drives. They are named differently than /dev/sda1 for example
if [[ "$DISK" == *"nvme"* ]]; then
        PREFIX="p"
fi

# Choose what label to make defined by the system type(UEFI or legacy)
if $UEFI; then
  parted -s "$DISK" mklabel gpt &> /dev/null
else
  parted -s "$DISK" mklabel msdos &> /dev/null
fi
echo "4" | tr -d '\n' > /tmp/progress
```

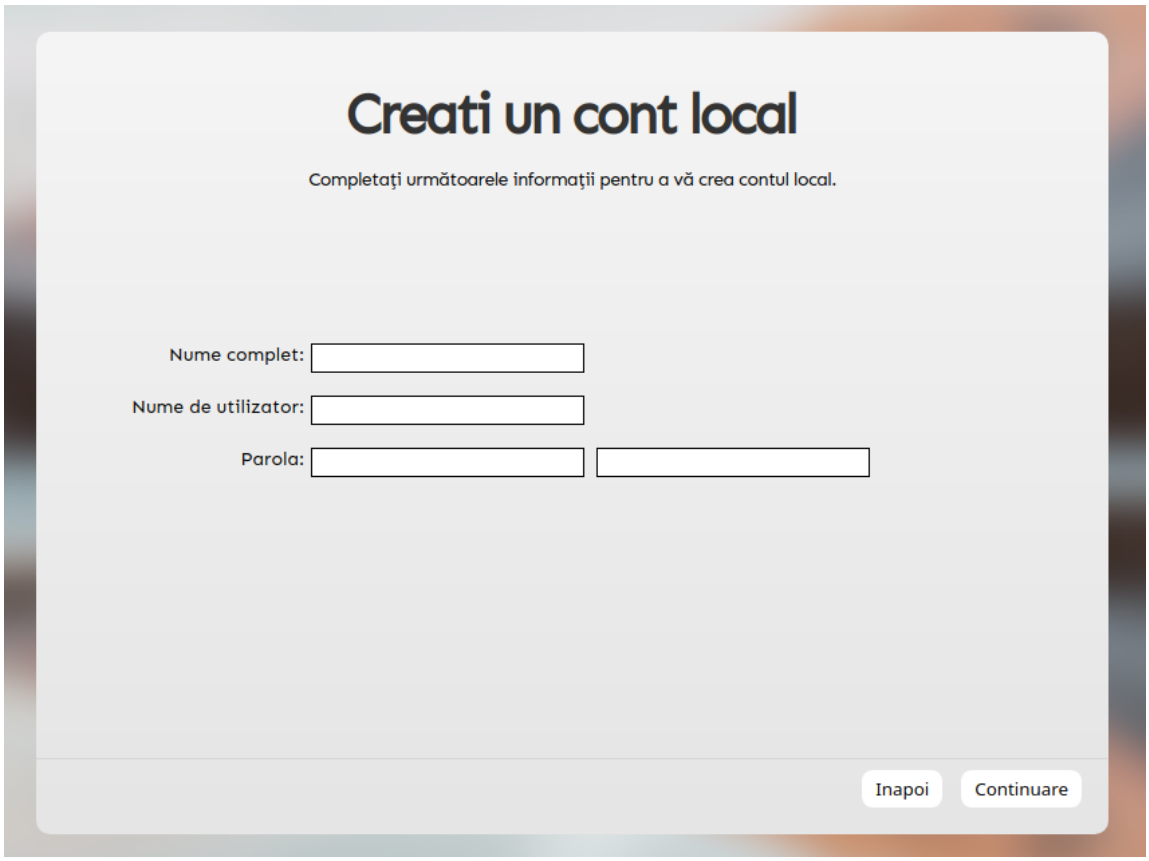
Iar apoi se procedeaza cu instalarea normala de sistem:

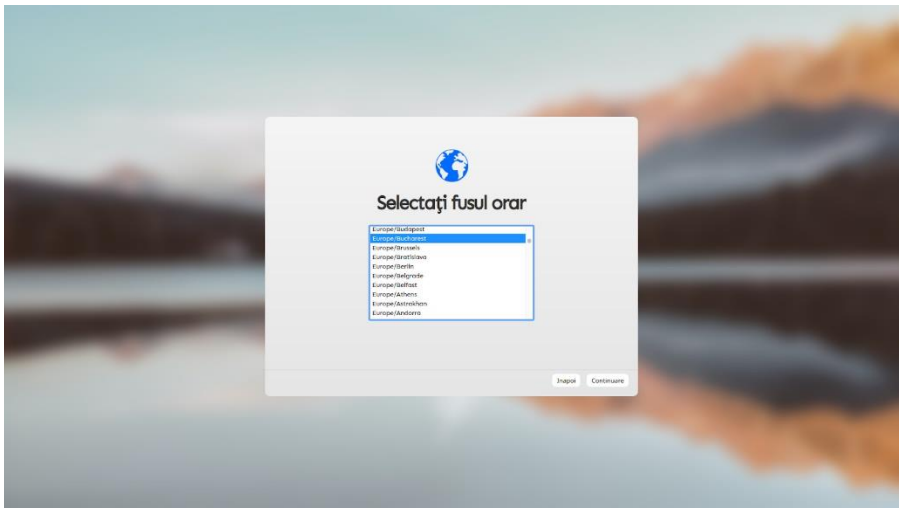
```
# adding pacman config directly
mkdir -p /mnt/etc/
cp /etc/pacman.conf /mnt/etc/pacman.conf
echo "17" | tr -d '\n' > /tmp/progress

# installing packages using pacstrap from the BASE_PACKAGES array variable
pacstrap /mnt "${BASE_PACKAGES[@]}"
echo "21" | tr -d '\n' > /tmp/progress

# Generating fstab
genfstab -U -p /mnt >> /mnt/etc/fstab
echo "58" | tr -d '\n' > /tmp/progress

# Pushing temporary Timezone to localtime file
ln -sf /usr/share/zoneinfo/"$ZONE" /mnt/etc/localtime
echo "60" | tr -d '\n' > /tmp/progress

# Enable or disable UTC time
if $utc_enabled; then
  arch-chroot /mnt hwclock --systohc --utc
else
  arch-chroot /mnt hwclock --systohc --localtime
fi
sed -i "s/#en_US.UTF-8/en_US.UTF-8/" /mnt/etc/locale.gen
echo "62" | tr -d '\n' > /tmp/progress

# Passing custom locale to locale file
if [ "$LOCALE" != "en_US.UTF-8" ]; then
  sed -i "s/#$LOCALE/$LOCALE/" /mnt/etc/locale.gen
fi

# making the language equal to the locale, and passing to locale.conf
echo "LANG=$LOCALE" > /mnt/etc/locale.conf
echo "64" | tr -d '\n' > /tmp/progress

# If there is a custom keymap, add it to the vconsole
if [ "$KEYMAP" != "us" ]; then
  echo "KEYMAP=$KEYMAP" > /mnt/etc/vconsole.conf
  if [ "$DESKTOP" != "None" ]; then
      echo -e "Section \"InputClass\"\nIdentifier \"system-keyboard\"\nMatchIsKeyboard \"on\"\nOption \"XkbLayout\" \"$KEYMAP\"\nEndSection" > /m
  fi
fi

# Comitting the locale settings
arch-chroot /mnt locale-gen &> /dev/null
echo "68" | tr -d '\n' > /tmp/progress

# Setting the hostname
echo "$HOST_NAME" > /mnt/etc/hostname
echo "70" | tr -d '\n' > /tmp/progress

# Adding localhost to hosts file
echo -e "127.0.0.1\tlocalhost\n::1\t\tlocalhost\n127.0.1.1\t$HOST_NAME.localdomain\t$HOST_NAME" >> /mnt/etc/hosts
echo "74" | tr -d '\n' > /tmp/progress
```

Dupa ce sistemul a fost instalat, necesita repornirea acestuia. Dupa repornire, aplicatia de User Setup porneste automat:

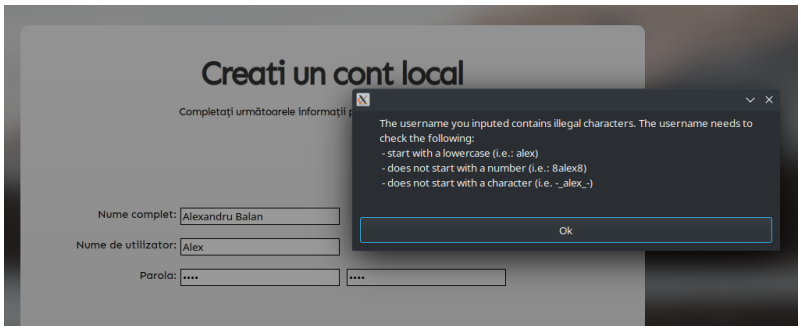


aceasta cerand date despre utilizator si despre limba/regiunea utilizatorului, fusul orar, keyboard layout-ul, numele complet, numele de utilizator si o parola.

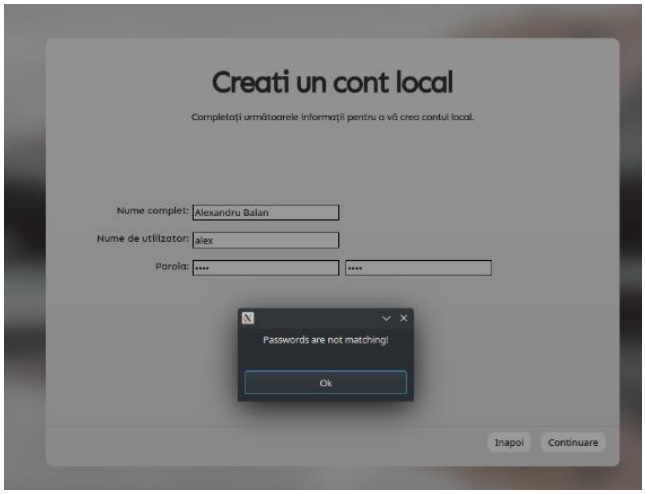




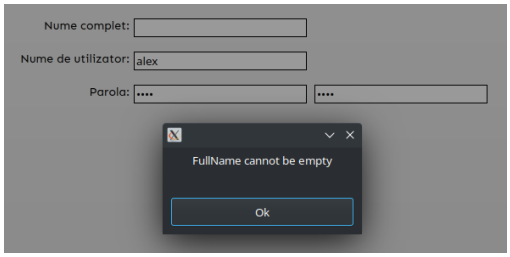
In caz ca utilizatorul introduce un nume de utilizator care nu este conform REFEX-ului, va primi o eroare in acest caz:



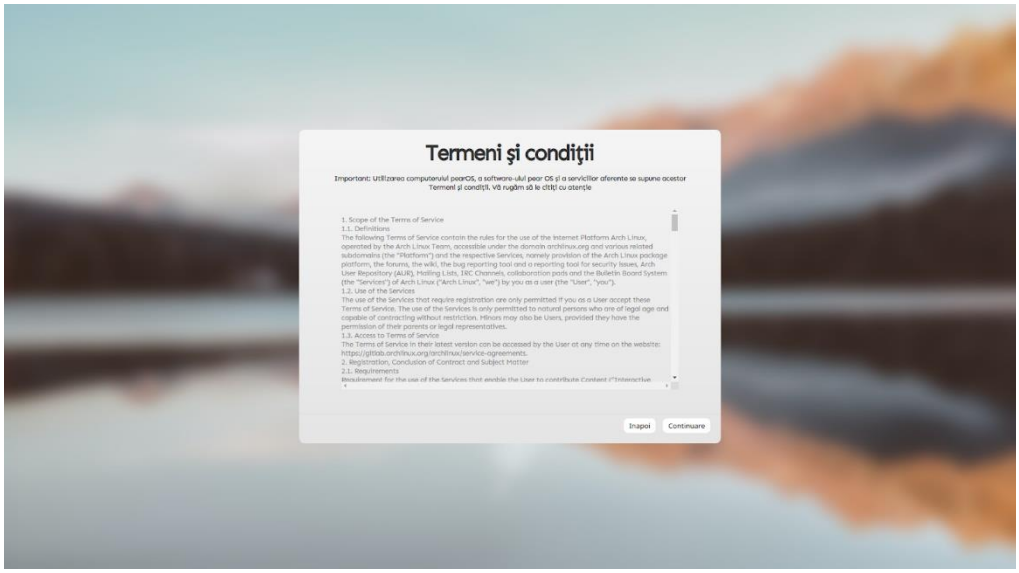
In caz ca utilizatorul greseste Password Confirm, acesta primeste o eroare in acest sens:



Daca utilizatorul uita sa completeze un camp, acesta primeste o eroare:



Dupa colectarea datelor referitoare la utilizator, apare fereastra cu Termenii si Conditiile:

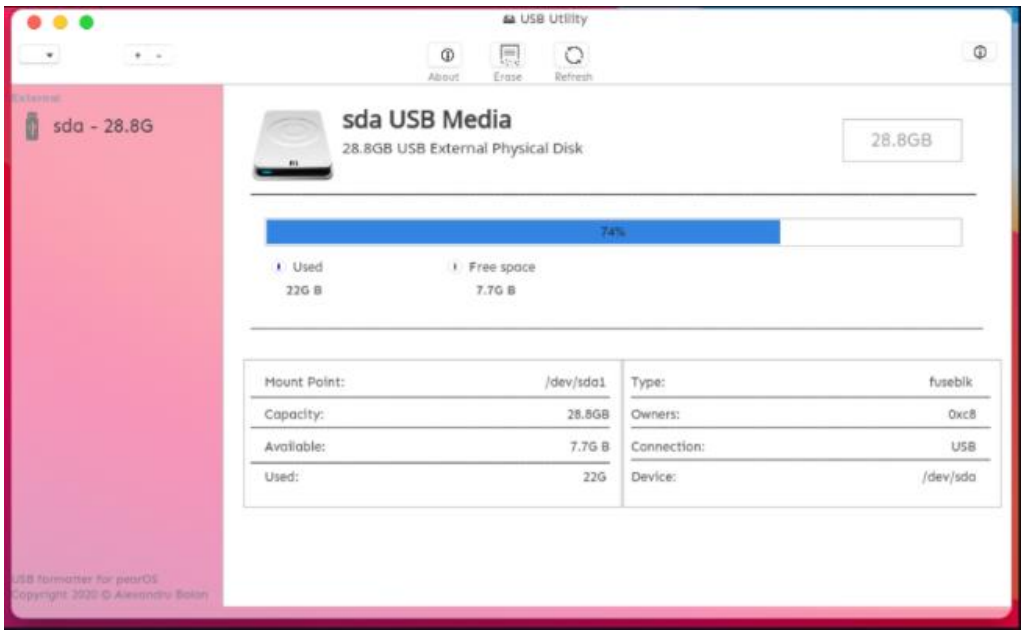


Imediat dupa acceptarea lor, instalarea finala incepe.
Codul sursa disponibil pe: *https://github.com/pearOS-archlinux/pearOS-installer*

Instalarea finala consta in creearea utilizatorului final, stergerea uilizatorului temporar, instalarea unor pachete de aplicatii necesare, stergerea uneor pachete de aplicatii inutile, care au fost folosite la momentul instalarii, spre exemplu `electron`, `nodejs`, si `npm`

O alta componenta (care e in lucru, urmeaza a fi introdusa in pearOS printr-un update viitor, este USB Utility:

- **USB Utility: cu care formatam stick-uri USB si carduri de memorie:**



Aplicația este făcută în totalitate de mine, în Gambas3, codul sursă fiind accesibil aici: https://github.com/Pear-Project/USBUtility

Aplicația lucrează intern în baza a mai multor funcții, scrise de mine:

- **Funcția care primește/afla partițiile/disk-urile valabile:**

```
'''''''''''''''''''''''''''''''''''''''''''''''Getting mountpoints function
Public Function GetMountPoints(sDrive As String) As String[]
Dim sMountPoints As New String[]
Dim sDriveItem As String
Dim x, y, z, a As Integer

  For x = 1 To sDrives.max
    If InStr(sDrives[x], sDrive & " ") = 1 Then
      If InStr(sDrives[x], " disk ") Then
        If InStr(sDrives[x], " /") Then
          a = InStr(sDrives[x], " disk ")
          sDriveItem = Mid$(sDrives[x], a + 6, Len(sDrives[x]) - (a + 5))
          sMountPoints.Add(sDriveItem)
          Return sMountPoints
        Endif
      Endif
      For y = (x + 1) To sDrives.max
        z = InStr(sDrives[y], sDrive)
        If z > 0 Then
          a = InStr(sDrives[y], " part ", z)
          sDriveItem = Mid$(sDrives[y], a + 6, Len(sDrives[y]) - (a + 5))
          sMountPoints.Add(sDriveItem)
        Endif
      Next
    Endif
  Next
  Return sMountPoints
End
```

- **Funcția care citește date despre disk-uri și le afișează în frontend:**

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''Launch data'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub GetSDDrives()
Dim sData, sLine, sUSB As String
Dim sDriveData, sUSBData As String[]
Dim usage As String
Dim exact As String
Dim avail As String
Dim mpoint As String
Dim fstypedata As String
  Shell "ls -l /dev/disk/by-path/*usb* | grep -v \"part\" | awk '{print $NF}' | awk -F \"/\" '{print $NF}'" To sUSB
  Shell "lsblk" To sData
  sUSBData = Split(sUSB, gb.NewLine)
  sDrives = Split(sData, gb.NewLine)
  For Each sLine In sDrives
    If sLine <> "" Then
      sDriveData = Split(sLine, " ", "", True)
      If sDriveData[5] = "disk" Then
        If sUSBData.Exist(sDriveData[0]) Then
            If sDriveData[4] = "0" Then
              cboSDCard.Add(sDriveData[0] & " - " & sDriveData[3])
              cboo.text = cboSDCard.text
              Label3.Text = (sDriveData[0] & " USB Media") '''''''''''''''''''working lmao
              Label4.Text = (sDriveData[3] & "B USB External Physical Disk")
              Label9.Text = (sDriveData[3] & "B")
              Shell "df -h --output=pcent /dev/" & (sDriveData[0]) & "1" & " | awk -F'%' 'NR==2{print $1}'" To usage
              Shell "df -h  --output=used /dev/" & (sDriveData[0]) & "1" & " | awk -F'%' 'NR==2{print $1}'" To exact
              Shell "df -h  --output=avail /dev/" & (sDriveData[0]) & "1" & " | awk -F'%' 'NR==2{print $1}'" To avail ''''''''''''''''''''
              Shell "df -h  --output=source /dev/" & (sDriveData[0]) & "1" & " | awk -F'%' 'NR==2{print $1}'" To mpoint
              Shell "df -h  --output=fstype /dev/" & (sDriveData[0]) & "1" & " | awk -F'%' 'NR==2{print $1}'" To fstypedata
              TextLabel4.text = exact & "B"
              TextLabel5.Text = avail & "B"
              lbl_capacity.Text = (sDriveData[3] & "B")
              lbl_mountpoint.Text = mpoint
              lbl_fstype.Text = fstypedata
              lbl_path.Text = "/dev/" & (sDriveData[0])
              lbl_available.Text = avail & "B"
              lbl_used.Text = exact
              Print Val(usage)
              pg_space.value = Val(usage) / 100
            Endif
        Endif
      Endif
    Endif
  Next
End
```

- **Functia care da refresh la disk urile disponibile:**

```
Public Sub RefreshSDDrives()
Dim sData As String
  Shell "lsblk" To sData
  sDrives = Split(sData, gb.NewLine)
End

Public Function GetPartitions(sDrive As String) As String[]
Dim sPartitions As New String[]
Dim sDriveItem As String
Dim x, y, z, a As Integer

  For x = 1 To sDrives.max
    If InStr(sDrives[x], sDrive & " ") = 1 Then
      For y = (x + 1) To sDrives.max
        z = InStr(sDrives[y], sDrive)
        If z > 0 Then
          a = InStr(sDrives[y], " ", z)
          sDriveItem = Mid$(sDrives[y], z, a - z)
          sPartitions.Add(sDriveItem)
        Endif
      Next
      If sPartitions.Count = 0 Then
        sPartitions.Add(sDrive)
      Endif
    Endif
  Next
  Return sPartitions
End
```

- **Funcția care verifica label-ul introdus pentru a nu insera caractere ilegale:**

```
Public Function isValidLabelName(sName As String) As Boolean
Dim sValid As String = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
Dim x As Integer

  For x = 1 To Len(sName)
    If Not InStr(sValid, Mid$(sName, x, 1)) Then
      Return False
    Endif
  Next
  Return True
End

Public Sub btnBegin_Click()
  dialogF.Show
End
```

- **Funcția care formateaza disk-ul selectat în filesystem-ul selectat:**

```
iIndex = Message.Question("Format Drive... Are You Sure?", "Yes", "No")
If iIndex <> 1 Then
  PictureBox1.Visible = True
  btnBegin.Enabled = True
  btnExit.Enabled = True
  Return
Endif
For iIndex = 0 To sPartitions.Max
  Shell "pkexec umount '/dev/" & sPartitions[iIndex] & "'" Wait
  Shell "pkexec rmdir '" & sMountPoints[iIndex] & "'" Wait
  If sPartitions[iIndex] <> "" Then
    Shell "pkexec /sbin/parted '/dev/" & sDrive & "' rm " & Right$(sPartitions[iIndex], Len(sPartitions[iIndex]) - Len(sDrive)) Wait
  Endif
Next
If sDiskLabel = "" Then
  sDiskLabel = "NewVolume"
Endif
```

Codul sursă de la componenta aceasta poate fi găsit pe https://github.com/Pear-Project/USBUtility

Actualizatorul de sistem:

Este realizat in Python(backend) si Glade(frontend). Se foloseste de package manager-ul de la Arch Linux, si anume pacman. Acesta verifica daca anumite pachete sunt sau nu invechite.

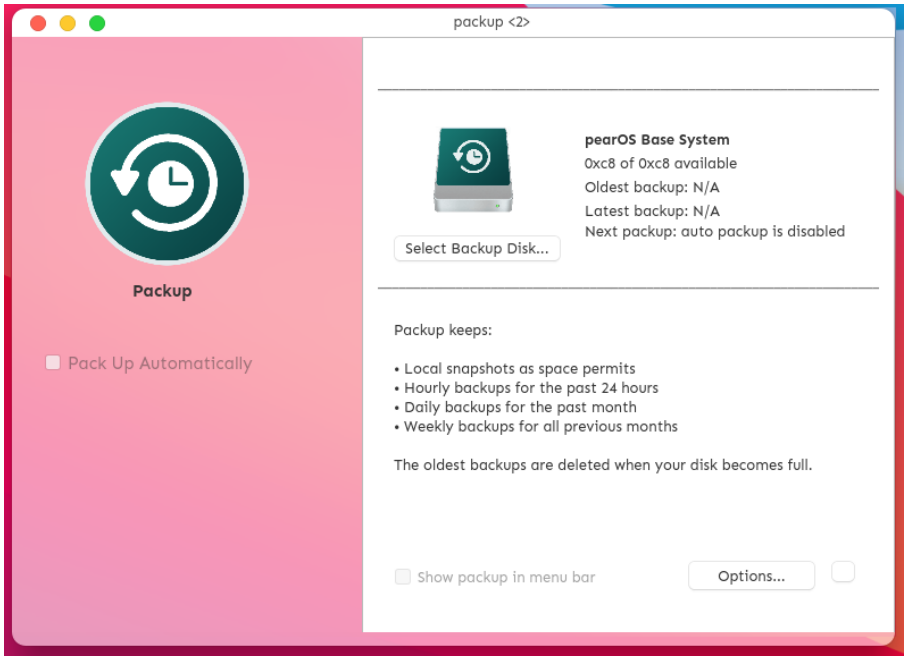
Aici poate fi azut codul care verifica actualizarile si trimite rezultatul la front-end:

```
def checkUpdate(self):
    updateLabel = self.builder.get_object("idLabel")
    updateName = self.builder.get_object("idUpdate")
    try:
        updates = subprocess.check_output("echo 'n' | pkexec pacman -Syu 2> /dev/null | grep 'system-update\|pearos-livecd-desktop\|pearos-settings\|filesys
        if updates != "":
            os.system("mkdir -p /tmp/system-update/ && wget -O /tmp/system-update/update.ini 192.168.1.2/update-info/update.ini")
            updateLabel.set_markup("<b>Updates are available for your Pear</b>")
            config = configparser.ConfigParser()
            config.sections()
            config.read('/tmp/system-update/update.ini')
            config.sections()
            nice = config['pext-header']['UpdateName']
            updateName.set_markup(nice)
        else:
            noup = subprocess.check_output("""echo -e "<b>Your Pear is up to date - pearOS NiceC0re</b>" | tr -d '\n' """, shell=True)
            noup = str(noup, 'utf-8')
            distro = subprocess.check_output("""cat /etc/os-release | egrep 'VARIANT' | cut -d = -f 2 | tr -d '\n"' """, shell=True)
            distro = str(distro, 'utf-8')
            label = " ".join((noup, distro))
            updateLabel.set_markup(label)
            updateButton = self.builder.get_object("updateButton")
            lblAgreement = self.builder.get_object("lblAgreement")
            updateButton.hide()
            lblAgreement.set_text("")

            config = configparser.ConfigParser()
            config.sections()
            config.read('/tmp/system-update/update.ini')
            config.sections()
            nice = subprocess.check_output("""date '+ Last checked: %A, %B %d, %Y at %H:%M'""", shell=True)
            nice = str(nice, 'utf-8')
            updateName.set_markup(nice)
```

Codul sursa disponibil pe: https://github.com/pearOS-archlinux/pext-installer

Packup

Intr-un update viitor voi include aplicatia care creeaza backup-uri, numita packup:



Aplicatia fiind dezvoltata in Gambas3 pentru Front-end si in Python pentru back-end.
Pentru backup-urile setate la o anumita perioada de timp, folosesc crontab, componenta valabina in Linux si aproximativ orice UNIX-like OS.

Codul sursa disponibil pe https://github.com/Pear-Project/packup

Am anexat un document README cu toate componentele care nu imi apartin si locuri de unde m-am inspirat(cod sursa, aplicatii, imagini)