T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

LİNUX DAĞITIMLARINDA MODERN UYGULAMA PAKET FORMATLARI İLE GELENEKSEL UYGULAMA PAKET FORMATLARININ KARŞILAŞTIRILMASI

ADNAN AKCAN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Prof. Dr. Gürkan TUNA

EDİRNE-2020

Adnan AKCAN'ın hazırladığı "Linux Dağıtımlarında Modern Uygulama Paket Formatları İle Geleneksel Uygulama Paket Formatlarının Karşılaştırılması" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Yüksek lisans tezi olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad): İmza

Prof. Dr. Gürkan TUNA .....

Dr. Öğr. Üyesi Tarık YERLİKAYA .....

Dr. Öğr. Üyesi Fatih AYDIN .....

Tez Savunma Tarihi: 13/01/2020

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Prof. Dr. Gürkan TUNA
Tez Danışmanı .....

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

.....
Prof. Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI YÜKSEK LİSANS PROGRAMI

DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

13/01/2020

Adnan AKCAN



Yüksek Lisans Tezi

Linux Dağıtımlarında Modern Uygulama Paket Formatları ile Geleneksel Uygulama Paket Formatlarının Karşılaştırılması

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

İşletim sistemleri arasında özel bir yere sahip olan Linux dağıtımlarında kullanılan geleneksel uygulama paket yöneticilerinin yerini son zamanlarda ortaya çıkan dağıtımdan bağımsız çalışabilen modern uygulama paket yöneticileri almaya başlamıştır. Bazı Linux dağıtımlarının içerisinde yerleşik olarak gelen bu paket yöneticileri günden güne büyümeye devam etmektedir. Linux kullanıcılarının oluşturduğu meraklı topluluğun önemli bir kısmı bu durumun Linux açık kaynak projesinin sonunu getireceğinden endişe duymaktadır. Kullanımı sürekli artan Linux dağıtımların paket yöneticilerinin geleceğinin şekillenmesinde bu üstünlük mücadelesinin etkili olacağı düşünülmektedir.

Geleneksel anlamda bilinen paket yöneticileri ile modern olan AppImage, Flatpak ve Snappy paket yöneticileri arasında belirgin farklar bulunmaktadır. Bunlardan en dikkat çekeni geleneksel paket bağımlılıkları haricinde modern paket yöneticilerinin sanal alan oluşturarak kendi bağımlılıklarını yaratmış olmasıdır. Ayrıca bunlar, genel olarak bazı sorunlara da sahiplerdir. Örneğin, AppImage formatının dağıtımlarda bir paket yöneticisi bulunmuyor, Flatpak diskte oldukça fazla alan kaplıyor ve Snappy tüm geliştiricilerin erişim sağlayıp uygulama paketlerine müdahale edebileceği bir ortamda bulunmuyor.

Modern paket formatlarının geleneksel olanlara karşın en büyük avantajı dağıtımdan bağımsız olarak çalışmalarıdır. Bunlar çoğu dağıtımda çalışmayı desteklemektedirler. AppImage aynı uygulamanın birden çok sürümünü kolaylıkla bir sistem üzerinde çalıştırabilir. Ayrıca uygulama çalıştırmak için herhangi bir kök iznine ihtiyaç duymaz.

Bu tez çalışmasında geleneksel paket formatı olarak Debian türevi olan Linux Mint ve Red Hat türevi olan Fedora dağıtımların varsayılan paket formatı seçilmiştir.

Geleneksel ve modern paket formatlarının önemli karakteristik özellikleri ve onların örnek uygulamalarının çalışma performansları incelenmiştir. Ayrıca kullanıcı açısından bu paket yöneticilerinin avantajları ve dezavantajları irdelenmektedir.

Yıl : 2020

Sayfa Sayısı : 87

Anahtar Kelimeler : deb, rpm, appimage, flatpak, snappy, paket yönetimi, paket bağımlılığı, sanal alan.

Master Thesis

Comparison of Modern Application Package Formats and Traditional Application Package Formats in Linux Distributions

Trakya University Institute of Natural Sciences

Computer Engineering Department

ABSTRACT

The traditional application package managers used in Linux distributions, which have a special place among the operating systems, have started to be replaced by modern application package managers that can work independently of the distribution recently. These package managers, which are built into some Linux distributions, continue to grow day by day. An important part of the curious community of Linux users is concerned that this will bring an end to the Linux open source project. It is thought that this struggle for superiority will be effective in shaping the future of package managers of Linux distributions whose usage is continuously increasing.

There are significant differences between traditionally known package managers and modern AppImage, Flatpak and Snappy package managers. The most notable of these is that, apart from the traditional package dependencies, modern package managers create their own dependencies by creating sandboxes. They also generally have some problems. For example, the AppImage format does not include a package manager in distributions, Flatpak takes up a lot of disk space, and Snappy is not in an environment where all developers can access and interfere with application packages.

The major advantage of modern package formats over traditional ones is that they work independently of distribution. They support running in most distributions. AppImage can easily run multiple versions of the same application on a system. It also does not need any root permissions to run the application.

In this thesis, the default package format of Debian derivative Linux Mint and Red Hat derivative Fedora distributions was chosen as the traditional package format. The important characteristics of the traditional and modern package formats and their performance of the sample applications are examined. Furthermore, the advantages and disadvantages of these package managers for the user are discussed.

Year : 2020

Number of Pages : 87

Keywords : deb, rpm, appimage, flatpak, snappy, package management, package dependency, sandbox.

TEŞEKKÜR

Tez çalışmam boyunca emeği geçen, bilgi ve tecrübeleriyle bana yol gösteren ve destek veren saygıdeğer danışman hocam Prof. Dr. Gürkan TUNA'ya tüm katkılarından dolayı teşekkür ederim.

Yüksek lisans öğrenimim boyunca destek olan ve anlayışlı davranan eşim Ayşe Gül AKCAN'a ve tüm öğrenim hayatım boyunca bana destek olan aileme teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ xi
ÇİZELGE LİSTESİ xiii
SİMGELER VE KISALTMALAR xiv
SİMGELER xiv
KISALTMALAR xv
1. GİRİŞ 1
2. LİNUX DAĞITIMLARI VE ÖZELLİKLERİ 8
2.1. Linux Çekirdeği 8
2.2. Linux Dağıtımları 9
2.3. Linux Dağıtımlarının Karakteristik Özelleri 12
3. DAĞITIMLARDA UYGULAMA PAKETLERİ 13
3.1. Yazılım Paketleri ve Paket Formatı 13
3.2. Paket Dönüştürme 13
3.3. Paket Bağımlılığı 16
3.4. Yazılım Paket Depoları 18
3.5. Kişisel Paket Arşivleri 20
3.6. Yazılım Deposu Kaynak Listesi 20
4. PAKET YÖNETİM SİSTEMLERİ 22
4.1. Paket Yönetim Sistemi 22
4.2. Birincil Paket Yöneticileri 24
4.2.1 Geleneksel Paket Yönetim Sistemleri ve Formatları 24
4.3. Çoklu Paket Yöneticileri 25
4.3.1 Modern Paket Yönetim Sistemleri ve Formatları 25
4.3.1.1 AppImage 27
4.3.1.2 Flatpak 28
4.3.1.3 Winepak 30
4.3.1.4 Snappy 30
4.3.2 Kapsayıcı Paket Yöneticileri 32
4.3.2.1 Sysget 32

4.3.2.2 Nix 32
4.3.2.3 GNU Guix 33
4.3.2.4 Nix ve GNU Guix Paket Yöneticilerinin Karşılaştırılması 34
5. UYGULAMALARIN SANAL ALANDA ÇALIŞMASI 35
5.1. Sanallaştırma 35
5.2. Konteyner 38
5.3. Sanal Makine Monitörü ve Konteyner Arasındaki Farklar 40
5.4. İşletim Sistemi İmajı 40
5.5. Sanal Alan 41
5.6. Chroot 41
5.7. SELinux 42
6. UYGULAMA PAKET FORMATLARININ KARŞILAŞTIRILMASI 44
6.1. Linux Mint Dağıtımında Örnek Uygulamaların Paket Formatları Açısından Karşılaştırılması 49
6.1.1. Vlc 50
6.1.2. Libreoffice 52
6.1.2.1. Libreoffice Uygulamasının Ağır Yük Altındaki Performansı 53
6.1.3. Kdenlive 54
6.2. Fedora Dağıtımında Örnek Uygulamaların Paket Formatları Açısından Karşılaştırılması 56
6.2.1. Vlc 57
6.2.2. Libreoffice 59
6.2.2.1. Libreoffice Uygulamasının Ağır Yük Altındaki Performansı 60
6.2.3. Kdenlive 61
7. SONUÇLAR VE TARTIŞMA 63
KAYNAKLAR 65
ÖZGEÇMİŞ 70

ŞEKİLLER LİSTESİ

Şekil 1.1. Dünya Çapında İşletim Sistemi Pazar Payı 1
Şekil 1.2. Bilgisayarların Eski ve Yeni Ürün Yazılımı Arayüzü 3
Şekil 1.3. Kişisel Bilgisayar Sistemin Genel Olarak Temel Bileşen Katmanları 4
Şekil 1.4. Unix ve GNU/Linux Gelişiminin Başlangıç Zaman Tüneli 6
Şekil 2.1. Linux Çekirdeğinin Bir Sistemdeki Yeri 9
Şekil 2.2. Bir Linux Dağıtımının Genel Mimari Yapısı 10
Şekil 2.3. Dünya Çapında Mobil İşletim Sistemi Pazar Payı 11
Şekil 3.1. Geleneksel Paket Formatları Dönüştürme Uygulaması 14
Şekil 3.2. Kdenlive Video Düzenleme Uygulamasının Bağımlılık Listesi 17
Şekil 3.3. Linux Mint 19.2 Dağıtımına Ait Yazılım Deposu Penceresi 19
Şekil 4.1. Bir Uygulamanın Paket Yönetim Sistemindeki Kurulum Adımları 23
Şekil 4.2. AppImage, Flatpak ve Snappy Paket Yönetim Sistemlerinin Logoları 26
Şekil 4.3. AppImage Uygulama Yapısı 27
Şekil 4.4. Flatpak Uygulama Yapısı 30
Şekil 4.5. Snappy Uygulama Yapısı 31
Şekil 5.1. Sanallaştırma Yapmadan Önceki ve Yapıldıktan Sonraki Sistem Yapısı 36
Şekil 5.2. Tip 1 ve Tip 2 Hypervisor Yapıları 36
Şekil 5.3. Bir Donanım Sistemi Üzerindeki Konteyner Yapısı 38
Şekil 5.4. Docker Konteyner Uygulamasının Bir Sistemdeki Yapısı 39
Şekil 5.5. Sanal Makine Monitörü ve Konteyner Arasındaki Yapı Farkı 40
Şekil 5.6. Uygulamanın Sanal Alanda Olmadan ve Sanal Alanda Çalışma Yapısı 41
Şekil 5.7. Chroot Uygulaması İle Yeni Dosya Sistemi Yapısı 42
Şekil 6.1. AppImage, Flatpak, Snappy ve deb Arama Trendleri 45
Şekil 6.2. Linux Mint Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Özellikleri 49
Şekil 6.3. Linux Mint Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Geekbench Performans Sonuçları 50
Şekil 6.4. Linux Mint Dağıtımında Vlc Uygulamasının Modern ve Geleneksel Paket Formatlarında Disk Boyutu 51

Şekil 6.5. Linux Mint Dağıtımında Libreoffice Uygulamasının Modern ve Geleneksel Paket Formatlarında Disk Boyutu ve İşlem Performansı 53
Şekil 6.6. Linux Mint Dağıtımında Libreoffice Uygulamasının Farklı Paket Formatlarında Ağır Yük Altında İşlem Performansı 54
Şekil 6.7. Linux Mint Dağıtımında Kdenlive Uygulamasının Modern ve Geleneksel Paket Formatlarında Disk Boyutu ve İşlem Performansı 55
Şekil 6.8. Fedora Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Özellikleri 56
Şekil 6.9. Fedora Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Geekbench Performans Sonuçları 57
Şekil 6.10. Fedora Dağıtımında Vlc Uygulamasının Modern ve Geleneksel Paket Formatlarında Disk Boyutu 58
Şekil 6.11. Fedora Dağıtımında Libreoffice Uygulamasının Modern ve Geleneksel Paket Formatlarında Disk Boyutu ve İşlem Performansı 60
Şekil 6.12. Fedora Dağıtımında Libreoffice Uygulamasının Farklı Paket Formatlarında Ağır Yük Altında İşlem Performansı 61
Şekil 6.13. Fedora Dağıtımında Kdenlive Uygulamasının Modern ve Geleneksel Paket Formatlarında Disk Boyutu ve İşlem Performansı 62

ÇİZELGE LİSTESİ

Çizelge 4.1. Geleneksel Paket Formatları ve Yönetici Araçları 23
Çizelge 4.2. Nix ve GNU Guix Paket Yönetici Araçlarının Karşılaştırılması 34
Çizelge 6.1. Modern ve Geleneksel Paket Formatlarının Önemli Özellik Bilgileri 47
Çizelge 6.2. Linux Mint Dağıtımında Vlc Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları 51
Çizelge 6.3. Linux Mint Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları 52
Çizelge 6.4. Linux Mint Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında Ağır Yük Altında İnceleme Sonuçları 54
Çizelge 6.5. Linux Mint Dağıtımında Kdenlive Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları 55
Çizelge 6.6. Fedora Dağıtımında Vlc Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları 58
Çizelge 6.7. Fedora Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları 59
Çizelge 6.8. Fedora Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında Ağır Yük Altında İnceleme Sonuçları 61
Çizelge 6.9. Fedora Dağıtımında Kdenlive Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları 62

SİMGELER VE KISALTMALAR

SİMGELER

apk : Android Yazılım Paketi
appimage : AppImage Dosya Formatı
apt : Komut Satırı Paket Yöneticisi Aracı
Btrfs : B-tree Dosya Sistemi
bz2 : Sıkıştırılmış Arşiv Dosyası
bzip2 : Sıkıştırılmış Arşiv Dosyası
C : Yüksek Seviyeli ve Genel Maksatlı Bir Programlama Dili
C++ : Nesne Yönelimli ve Yüksek Seviyeli, Genel Maksatlı Programlama Dili
click : Ubuntu Akıllı Telefon Yazılım Paketi
deb : Debian Yazılım Paketi
dnf : Dandified Yum
dpkg : Debian Paket Yöneticisi Aracı
ext2 : İkinci Genişletilmiş Dosya Sistemi
ext3 : Üçüncü Genişletilmiş Dosya Sistemi
ext4 : Dördüncü Genişletilmiş Dosya Sistemi
flatpakref : Flatpak Dosya Formatı
GB : Gigabyte
git : Sürüm Kontrol Sistemi
GitHub : Web Tabanlı Bir Depolama Servisi
gzip : Sıkıştırılmış Arşiv Dosyası
init : Süreç Kontrol Başlangıcı
jfs : Günlüklü Dosya Sistemi
libc : GNU C Kütüphanesi
libostree : Uygulama Yürütme ve Yönetme Sistemi
Ltd : Limited
MB : Megabyte

mov : QuickTime Medya Dosyası
mp4 : MPEG-4 Bölüm 14 Dosya Formatı
OSTree : Uygulama Yürütme ve Yönetme Sistemi
pet : Puppy Yazılım Paketi
pisi : Pardus Eski Yazılım Paketi
pptx : Microsoft PowerPoint Dosyası
ReiserFS : Günlüklü Dosya Sistemi
rpm : Red Hat Yazılım Paketi
sn : Saniye
snap : Snappy Dosya Formatı
tar.bz2 : Sıkıştırılmış Arşiv Dosyası
tar.gz : Sıkıştırılmış Arşiv Dosyası
tbz2 : Sıkıştırılmış Arşiv Dosyası
tgz : Sıkıştırılmış Arşiv Dosyası
txz : Sıkıştırılmış Arşiv Dosyası
v : Versiyon
X11 : Pencere Sistemi
XFS : Yüksek Performanslı 64 bit Günlüklü Dosya Sistemi
yum : Fedora Yazılım Paketi
zypper : Komut Satırı Paket Yöneticisi Aracı

KISALTMALAR

ABD : Amerika Birleşik Devletleri
ALGOL : ALGOritmic Language(Yüksek Düzey Programlama Dili)
AT&T : American Telephone and Telegraph Company(Amerikan Telefon ve Telgraf Şirketi)
BASH : Bourne Again Shell(Komut Dili Yorumlayıcı)
BIOS : Basic Input Output System(Temel Giriş Çıkış Sistemi)
BSD : Berkeley Software Distribution(Berkeley Yazılım Dağıtımı)
CLI : Command Line Interface(Komut Satırı Arayüzü)
CVS : Concurrent Versions System(Sürüm Kontrol Sistemi)
DAC : Discretionary Access Control(İsteğe Bağlı Erişim Kontrolü)

DLL : Dynamic Link Library(Dinamik Link Kütüphanesi)

DPI : Dot Per Inch(İnç Başına Nokta)

GNOME : GNU Network Object Model Environment(GNU Ağ Nesne Modeli Ortamı)

GNU : GNU's Not Unix(GNU Unix Değildir)

GPL : General Public License(Genel Kamu Lisansı)

GTK : Gimp Toolkit(Çoklu Platform Geliştirme Araç Takımı)

GUI : Graphical User Interface(Grafiksel Kullanıcı Arayüzü)

GUILE : Ubiquitous Intelligent Language for Extensions(Genişletmeler için Her Yerde Akıllı Dil)

HEVC : High Effiency Video Coding(Yüksek Verimli Video Kodlama)

IOT : Internet of Things(Nesnelerin İnterneti)

ISO : International Organization for Standardization(Uluslararası Standartlar Örgütü)

KDE : K Desktop Environment(K Masaüstü Ortamı)

LISP : Locator/Identifier Separation Protocol(Konum Belirleyici / Tanımlayıcı Ayırma Protokolü)

LTS : Long Term Support(Uzun Süreli Destek)

LXC : Linux Containers(Linux Konteynerleri)

MAC : Mandatory Access Control(Zorunlu Erişim Kontrolü)

MIT : Massachusetts Institute of Technology(Massachusetts Teknoloji Enstitüsü)

MPEG : Motion Picture Experts Group(Hareketli Resim Uzmanları Grubu)

NSA : National Security Agency(Ulusal Güvenlik Ajansı)

OS : Operating System(İşletim Sistemi)

PDF : Portable Document Format(Taşınabilir Belge Biçimi)

PPA : Personal Package Archives(Kişisel Paket Arşivleri)

RAM : Random Access Memory(Rastgele Erişimli Bellek)

SD : System Distribution(Sistem Dağıtımı)

SELinux : Security-Enhanced Linux(Güvenlik Geliştirilmiş Linux)

UEFI : Unified Extensible Firmware Interface(Birleşik Genişletilebilir Ürün Yazılımı)

URL : Uniform Resource Locator(Standart Kaynak Bulucu)

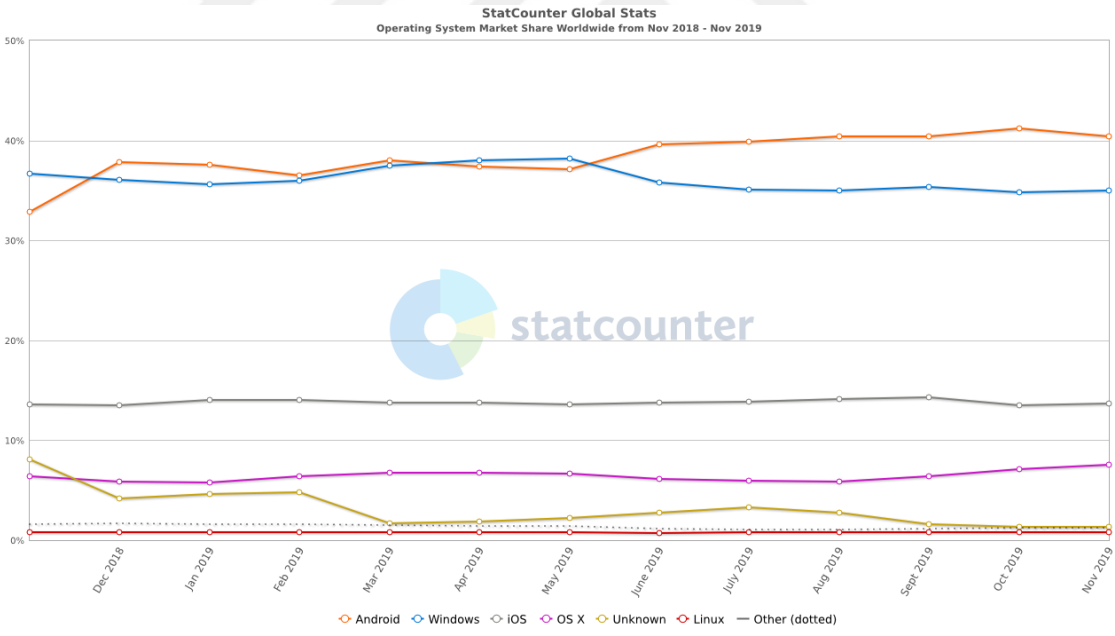
VMM : Virtual Machine Monitor(Sanal Makine Monitörü)

WINE : Wine Is Not an Emulator (Wine Öykünücü Değildir)

BÖLÜM 1

GİRİŞ

Günümüzde işletim sistemleri akıllı cep telefonlarından süper bilgisayarlara, elektrikli ev aletlerinden sürücüsüz arabalara kadar hayatımızın birçok alanında yer almaktadır ve bunların arasında en yaygın olarak akıllı cep telefonlarından sonra kişisel bilgisayarlarda kullanılmaktadır. Şekil 1.1'de dünya genelinde bütün platformlardaki işletim sistemleri kullanım oranları görünmektedir. Burada Linux işletim sisteminin payı az olmasına rağmen dünya genelinde özgün bir topluluk tarafından hem kullanılmakta hem de geliştirilmektedir.



Şekil 1.1 Dünya Çapında İşletim Sistemi Pazar Payı(StatCounter Global Stats, 2019).

İşletim sistemleri, kullanıcılar ile donanım arasında bir arayüz oluşturarak kullanıcıların ihtiyaçları doğrultusunda giriş ve çıkış aygıtlarını işleyerek onlara birçok hizmet sunan esnek yazılım setleridir. Bir işletim sistemi kullanıcısı, sistemin ve donanımın izin verdiği ölçüde bilgisayarına istediği işlemleri yaptırabilir ve bunun için

işletim sistemi dışında, bu sistemle uyumlu çalışan farklı uygulama yazılımlarından faydalanabilir (Tanenbaum & Bos, 2014).

İşletim sistemi, kullanıldığı cihaz üzerinde hemen hemen her şeyi bir arada tutan ve bunları yöneten çekirdek yazılım kümesidir ve donanıma en yakın yazılım birimi olarak cihaz donanımıyla iletişim halindedir. Bununla birlikte birçok cihaz, genellikle doğrudan cihazların hafızasına programlanan bir tür düşük seviye yazılım sayılan ürün yazılımı (firmware) çalıştırır. Ürün yazılımı genellikle sabit görevleri yerine getirmek için tasarlanmış küçük bir yazılım paketidir. Bu yazılım paketleri birçok elektronik cihazın içerisinde bulunmaktadır. Örneğin başta bilgisayarlar olmak üzere modemler, tabletler, müzik çalarlar ve mobil cihazlarda yer almaktadır (Catsoulis, 2009).

Bir televizyonun uzaktan kumandası ürün yazılımı içeren mühendislik ürünü örneğidir. Kumanda üzerindeki düğmelere basıldığında televizyon seti bu sinyalleri algılar ve işler. Bu cihazlar ürün yazılımı vasıtasıyla işlevlerini doğru bir şekilde yerine getirerek kararlı bir şekilde çalışmalarına devam etmektedirler (Brown, Eckhouse & Estabrook, 1977).

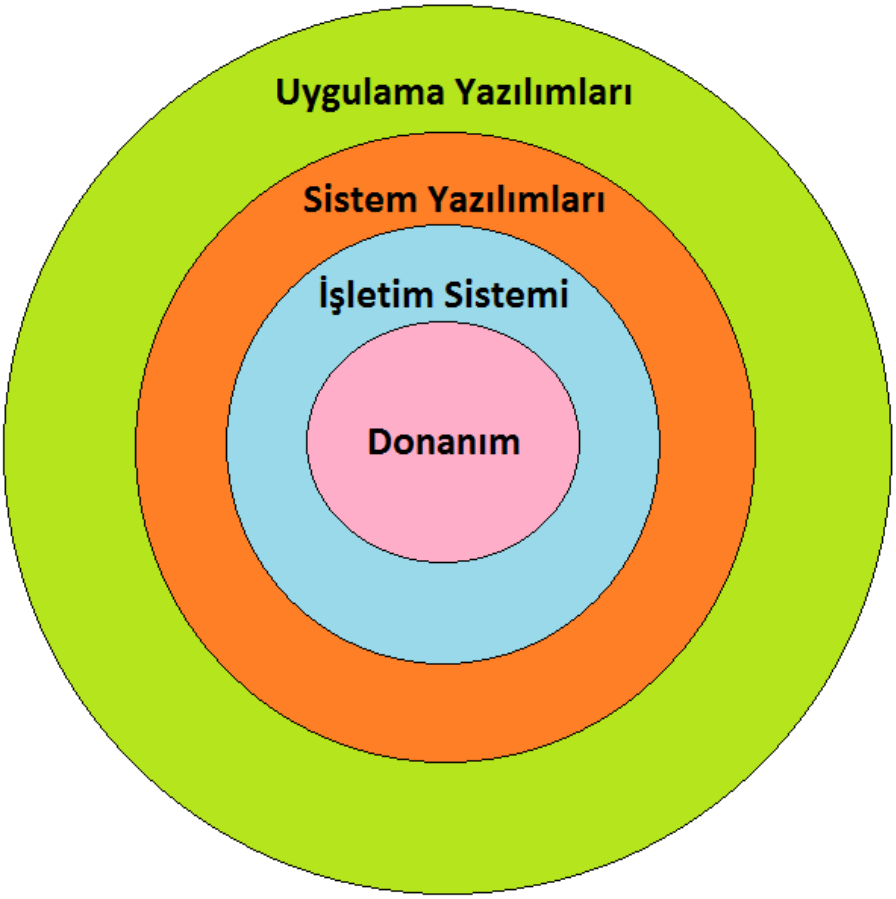
Modern bir kişisel bilgisayar çalışmaya başladığında, artık kullanımı eskimiş normal Temel Giriş Çıkış Sistemi'nden (BIOS) çok daha gelişmiş bir görünüme sahip olan Birleşik Genişletilebilir Ürün Yazılımı (UEFI) bilgisayarın anakartından sistem belleğine yüklenir. Şekil 1.2(b)'de gösterildiği gibi bilgisayar ürün yazılımı arayüzleri son zamanlarda neredeyse küçük bir işletim sistemi arayüzü haline dönüşmüştür. Ürün yazılımı, bilgisayarın donanımını hızla başlatan düşük seviye bir yazılımdır ve ardından işletim sistemini bilgisayarın sabit diskinden sistem belleğine önyükleme yapar. Bilgisayar açıldığında ürün yazılımı → çekirdek → tam işletim sistemi sırasıyla işlevlerini yerine getirir. Böylece, kişisel bir bilgisayarda bir işletim sistemi çalıştırabilmek için ürün yazılımına ihtiyaç duyulmaktadır (Brown vd., 1977).



(a) (b)

Şekil 1.2. Bilgisayarların Eski(a) ve Yeni(b) Ürün Yazılımı Arayüzü (Award Software,2019 & ASUS, 2019).

Bir kişisel bilgisayar sisteminin genel olarak donanım, işletim sistemi, sistem yazılımları ve uygulama yazılımları gibi temel bileşenleri olduğu gibi, işletim sisteminin de temel bileşeni çekirdek (kernel)'tir. Bu çekirdek giriş çıkış işlemlerinin yönetilmesi, işlem yönetimi, hafıza yönetimi, aygıt yönetimi, dosya yönetimi ve işletim sistemi için daha birçok temel görevin yerine getirilmesinden sorumludur. Sistem açılırken belleğe ilk önce yüklenenlerdendir ve sistem kapatılıncaya kadar ana bellekte kalır. Günümüzde son kullanıcı gözünden bakıldığında ne sadece bir donanım ne de sadece bir işletim sistemi bir şey ifade etmez. Şekil 1.3 gösterildiği gibi donanım ve yazılım birlikte çalışmalıdır.



Şekil 1.3. Kişisel Bilgisayar Sistemin Genel Olarak Temel Bileşen Katmanları.

Çekirdek genellikle bilgisayardaki başka bir yazılım tarafından değiştirilmemesi için izole bir alanda çalıştırılır ve imtiyazlı bir işlem düzeyi olan çekirdek kipi(Kernel Mode)'nde çalışır. Bu işlem düzeyinde çalışan yazılım, donanıma ve sistemin bütününe tam yetkili bir şekilde müdahalede bulunabilir. Uygulama yazılımları ise, kullanıcı kipi(User Mode) denilen daha sınırlandırılmış bir işlem düzeyinde çalışır. Kullanıcı kipinde çalışan yazılımlar, çekirdeğin ya da başka uygulamaların bellek alanlarına erişim sağlayamazlar, bazı sınırlandırılmış işlemci komutlarını çalıştıramazlar ve donanım kaynaklarına doğrudan müdahalede bulunamazlar (Dike, 2001).

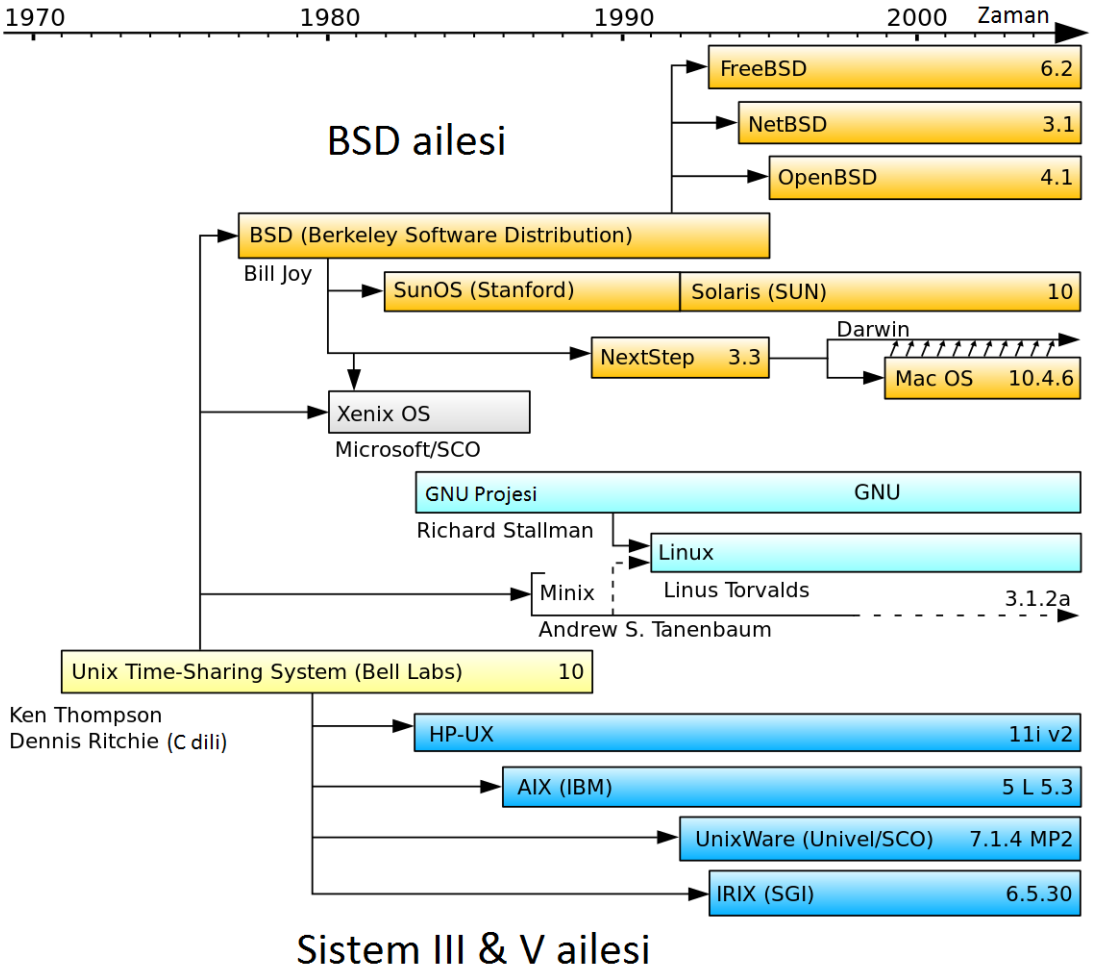
Kişisel bilgisayarlarda kullanılan en yaygın işletim sistemlerine bakıldığında pazar payına göre Microsoft Windows, Apple Mac İşletim Sistemi (OS) ve Linux çekirdeğini temel alan Linux dağıtımları gelmektedir. Bunların arasından ilk ikisi ücretli ve kapalı kaynak kodlu işletim sistemleri olmasına karşın bir GNU Unix Değildir (GNU) projesi olan Linux dağıtımları, serbestçe dağıtılabilen, çok görevli, çok

kullanıcılı, birçok alanda çalışabilen ve herhangi bir edinme maliyeti olmayan bir Unix işletim sistemi türevidir (Nandhini, Nivetha & Shobana, 2016).

1960'lı yıllarda Amerikan Telefon ve Telgraf Şirketi'nin (AT&T) Bell laboratuvarları, Massachusetts Teknoloji Enstitüsü (MIT) ve General Electric şirketi ortak olarak geliştirdikleri bir projede "Multics" adında bir işletim sisteminin patentini aldılar. Projeden Bell laboratuvarı çekildikten sonra proje çalışanları olan Dennis Ritchie ve Ken Thompson "Multics" işletim sisteminden esinlenerek "Unics" adında bir işletim sistemi projesine girdiler. Daha sonra bu proje Unix adıyla anıldı. O zamana kadar oluşturulan işletim sistemleri makine diliyle yazılmaktaydı. Bunun yerine Dennis Ritchie'nin katkılarıyla bu projenin C programlama dilinde yazılması işletim sisteminin diğer platformlara esnek bir şekilde taşınmasını sağladı. Oluşturulan işletim sistemi daha sonra özellikle başta üniversiteler olmak üzere akademik alanda sesini duyurmaya başladı (Spinellis, 2017).

1980'lere gelindiğinde AT&T Unix işletim sistemini ticari olarak özel lisanslar ile pazarlamaya başlaması ile birlikte Unix'in doğduğu günden itibaren gelişmesine katkıda bulunan çoğu kişinin bu duruma karşı çıkmasıyla Richard Stallman önderliğinde Unix benzeri ve ücretsiz dağıtılabilen bir işletim sistemi meydana getirmek için GNU projesi başlatıldı. Daha sonra ise 1987 yılında Prof. Andrew S. Tanenbaum akademik çalışmaları sırasında öğrencileri ile birlikte mikro çekirdek mimarili bir Unix türevi olan "Minix" işletim sistemini ortaya çıkardı (Spinellis, 2017).

Bir üniversite öğrencisi olan Linus Benedict Torwalds Prof. Andrew S. Tanenbaum'un geliştirmiş olduğu "Minix" işletim sistemini ile Richard Stallman'ın GNU projesini kendisinin oluşturduğu işletim sistemi çekirdeği ile birleştirerek oluşturduğu bütünleşik işletim sistemini Linux ismiyle 5 Ekim 1991 yılında 0.02 sürüm numarasıyla yayınlandı (Salus, 1994). Şekil 1.4'de Unix ve GNU/Linux ortaya çıkışının zamana göre gelişimi gösterilmektedir.



Şekil 1.4. Unix ve GNU/Linux Gelişiminin Başlangıç Zaman Tüneli (GNU/Linux Application Programming, 2008).

Dünya genelinde kullanılan onlarca hatta yüzlerce Linux çekirdeğinden türetilmiş dağıtım bulunmaktadır. Bir Linux dağıtımı, genellikle bu yazıda dağıtım olarak bahsedilecek ve işletim sistemi kavramının yerine de geçebilecek, kısaca Linux çekirdeğine dayanan bir takım yazılım araçları ile donatılmış açık kaynak kodlu bir yazılım paketleri bütünüdür (Mongkolluksame vd, 2012).

Linux dağıtımlarının hepsi tek bir ortak işletim sistemi çekirdeği kullanmasına rağmen Linux işletim sisteminin açık kaynak kodlu olması, bu sistemin hiçbir zaman bir grup ya da kişiye ait olmaması, Linux'un kullanıcılara vermiş olduğu özgür bir şekilde sistemi değiştirebilme kabiliyetiyle kullanıcıların kendi irade ve isteklerine göre işletim sistemini özelleştirmeleri birçok dağıtımın ortaya çıkmasına neden oldu (GNU Project - Free Software Foundation, 1989). Bu durum da farklı dağıtımlar için farklı paket

yönetim sistemleri ve paket formatları oluşturulmasına sebep oldu. Bu nedenle Linux dünyasında dağıtımları tek bir çatı altında birleştirmek zordur.

Her dağıtımın ayrı ya da bazılarının ortak paket yöneticileri kullanmasının yanında bazı geliştiriciler çoğu dağıtımda kendi temel paket yöneticisinden ayrı olarak çalışabilecek AppImage, Flatpak ve Snappy gibi yeni paket yöneticileri oluşturmuşlardır. Bu geliştiriciler kullanıcıların uygulamaları daha kolay bir şekilde kullanmasını amaçlamaktadırlar. Bu çalışmada modern paket yöneticileri olan AppImage, Flatpak ve Snappy paket formatları ile geleneksel olarak nitelenen Debian türevi dağıtımların varsayılan paket yöneticisi paket formatı olan deb'in önemli karakteristik özellikleri, kendi uygulamalarının çalışma performansları ve kullanıcı açısından bu paket yöneticilerinin avantajları ve dezavantajları incelenmiştir.

Bu tez çalışmasının metin kısmının oluşturulmasında ulusal ve uluslararası akademik kaynaklar, Ulusal Tez Merkezinden ulaşılabilen yüksek lisans ve doktora tezleri, veri tabanlarından ulaşılabilen makale ve bildiriler, ilgili kurum ve kuruluşlara ait yayımlanan tez konusu ile ilgili raporlar ve İnternet'ten elde edilen bilgilerden yararlanılmıştır.

Yapılan araştırma, bilişim sistemleri alanında önemi git gide artan Linux işletim sisteminin kullanım ve geliştirme yöntemlerinin belirlenmesi bakımından önemlidir. Bu tez çalışması yararlanılan kaynaklardan elde edilen bilgiler ve karşılaştırma testleri sonuçları ile derlenmiş bir bilimsel belge oluşturmak amacıyla hazırlanmıştır. Çalışmada nicel ve nitel gözlem yöntemi ile Debian türevi olan Linux Mint 19.2 ve Red Hat türevi olan Fedora 29.1.2 dağıtımı üzerinde uygulama testleri gerçekleştirilmiştir.

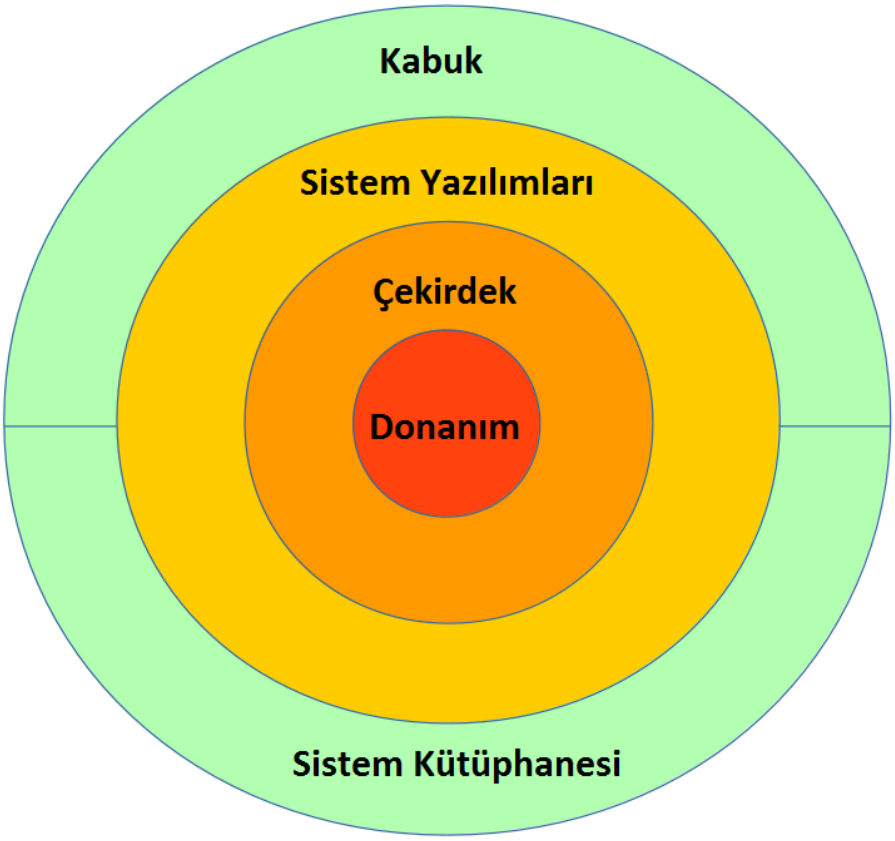
BÖLÜM 2

LİNUX DAĞITIMLARI VE ÖZELLİKLERİ

2.1. Linux Çekirdeği

Linux'un tarihi 1991'de Finlandiyalı bir öğrenci Linus Torvalds tarafından yeni bir ücretsiz işletim sistemi çekirdeği oluşturmak için kişisel bir projenin başlamasıyla başlamıştır. O zamandan beri, ortaya çıkan Linux Çekirdeği tarih boyunca sürekli bir büyüme göstererek çekirdekten türetilen birçok dağıtımın içerisinde yer almıştır. Linux Çekirdeği bir GNU Genel Kamu Lisansı (GPL) lisansına sahip bir açık kaynak projesidir, bu nedenle kaynağın yeniden dağıtılması ve üzerinde bazı yasal kısıtlamalar olmadan değişiklik yapılması mümkündür. Dünyanın her yerinden geliştiriciler bu çekirdeğe katkıda bulunmaktadırlar (Gonzalez-Barahona vd, 2005).

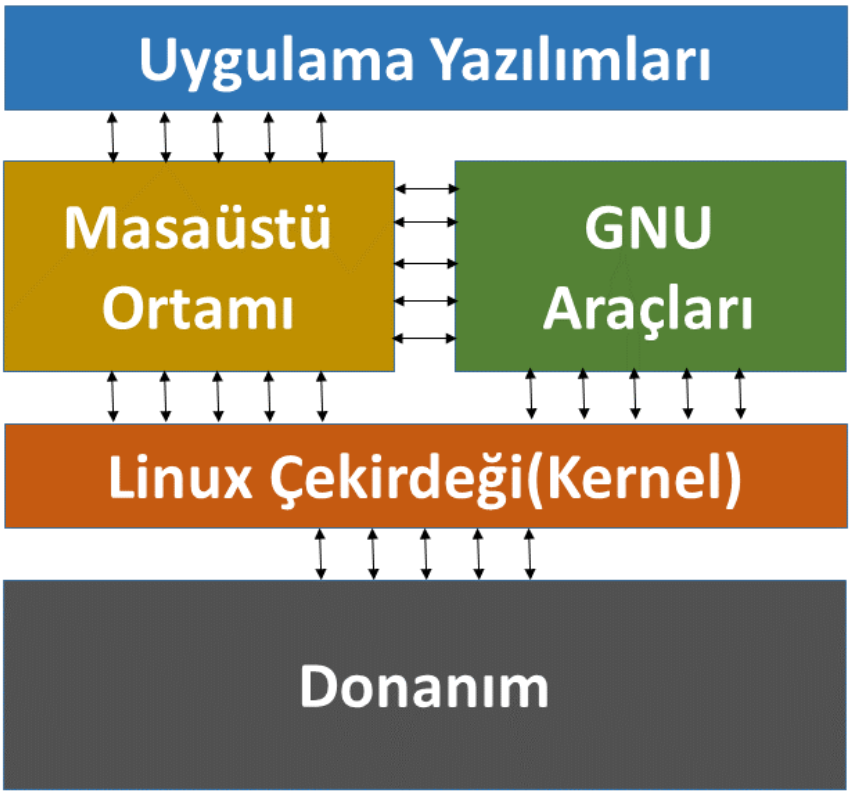
Linux çekirdeği, 1960'lı yıllarda geliştirilmiş, çoklu kullanıcılı, çoklu görev özelliği olan, sunucu, masaüstü ve dizüstü bilgisayarlarda çalışabilen, oldukça kararlı bir işletim sistemi olan Unix benzeri açık kaynak bir monolitik işletim sistemi çekirdeğidir. Linux işletim sistemi mimarisi temel olarak çekirdek, donanım katmanı, sistem kütüphanesi, kabuk (shell) ve sistem yardımcı programından oluşur. Bunlar Şekil 2.1'de de açıkça gösterilmektedir.



Şekil 2.1. Linux Çekirdeğinin Bir Sistemdeki Yeri.

2.2. Linux Dağıtımları

Bir Linux dağıtımı bir donanım üzerinde çalışan Linux çekirdeği, GNU araçları, masaüstü ortamı ve uygulama yazılımları ile çoğu aygıt sürücülerini içeren açık kaynak kodlu bir yazılım paketleri bütünüdür. Linux dağıtımı, Şekil 2.2'de gösterildiği gibi Linux çekirdeği ile birlikte yetenekleri birçok kullanıcının gereksinimlerini karşılayacak şekilde paketlenmiş çeşitli GNU araçları ve kütüphanelerini de içermektedir. Şekildeki oklar veri akışını göstermektedir.



Şekil 2.2. Bir Linux Dağıtımının Genel Mimari Yapısı.

Onlarca farklı çeşidi bulunan bu dağıtımlar masaüstü, dizüstü ve tablet bilgisayarlarda, akıllı cep telefonları, televizyonlar ve kol saatlerinde, web ve veri tabanı sunucuları ile daha birçok alanda kullanılmaktadır ve en son 277 tanesi popüler sıralamasına göre DistroWatch web sitesinde verilmektedir (DistroWatch, 2019). Debian, Slackware, Red Hat ve daha sonra Gentoo dağıtımların atası sayılmaktadır.

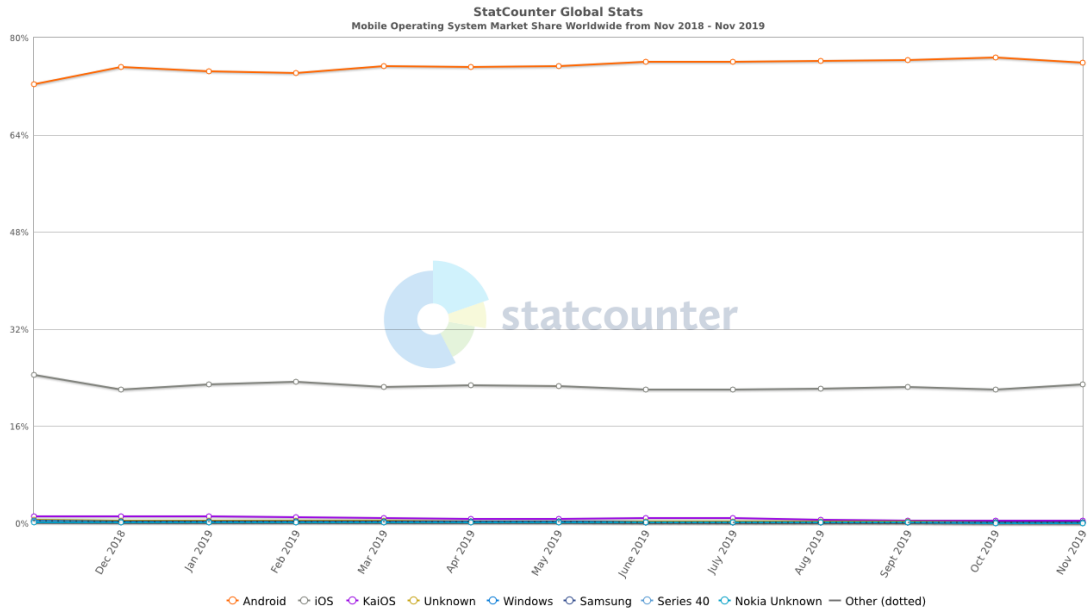
Çekirdeğin dağıtıldığı GPL lisansı sayesinde her tür gömülü sistemler, tabletler veya elektronik cihazlar oluşturmaya karar veren şirketler, mevcut dağıtımları çok kolay bir şekilde değiştirebilir ve yeniden dağıtabilirler. Bu avantaj özel kod geliştirme maliyetini tamamen ortadan kaldırabilir ve eğer kodun çok az bir şekilde yeniden kullanılmasından kaynaklanan maliyetler sayılmaz ise kurulan sistemler üzerindeki bakım maliyetlerini ciddi oranlarda düşürebilir (Dawson, DeWalt & Cleveland, 2016). Linux çekirdeğinin özel olarak lisanslandığı GPL ikinci sürüm, geliştiricilerin çekirdekte yaptıkları ve diğer işletim sistemlerinin düştüğü tuzakların çoğundan kaçınmada hayati önem taşıyan değişiklikleri kamuoyu önünde paylaşmalarını gerektirir. Geliştiricileri çekirdek yazılımını kullanmaya ve geliştirilmesine katkıda

bulunmaya teşvik eder, çünkü onlara hiç kimsenin kodlarını alıp kendi ürününde kapalı olarak kullanmamasını da sağlar (Zimmerman, 2016).

Yaygın kullanılan arama motorlarından birisi olan Google, Linux yapısını kullanan en büyük sistemlerden birisidir ve altyapısında çok miktarda Linux tabanlı sunucu çalışmaktadır. Google, Linux işletim sisteminin esnekliğini ve uygulanabilirliğini gösterme bakımından çok iyi bir örnektir (Tuna, 2008).

2003 yılında, Andy Rubin, Linux çekirdeğini temel alan Android işletim sistemini geliştirmeye başladı (Bazard & Bhardwaj, 2011). Android şirketini kurdu ve daha sonra şirket Google tarafından 2005 yılında satın alındı ve 2007'de açık kaynak lisansı ile (Apache 2.0) piyasaya sürüldü. Google, Open Handset Alliance ve özgür yazılım topluluğu tarafından geliştirilen Android GNU / Linux dağıtımlarından farklı olarak kullandığı Linux çekirdeği GPL, diğer harici parçalar ise Apache lisansı ile dağıtılmaktadır (Krajci & Cummings, 2013). Bu yapı, geliştiricileri Android'e katkı sağlamaları açısından teşvik ettiğinden devamlı olarak gelişmesini ve yeni özelliklerin eklenmesini sağlamıştır.

Doğrudan satışlar ve mobil pazarın baskınlığı dikkate alındığında, kullanıcı tercihleri Şekil 2.3'de görüldüğü gibi Linux çekirdeğini tüketici cihazları arasında en çok kullanılan çekirdeğe ve dolayısıyla dünyadaki en yaygın çekirdeğe dönüştürmüştür.



Şekil 2.3. Dünya Çapında Mobil İşletim Sistemi Pazar Payı (StatCounter G. Stats, 2019).

2.3. Linux Dağıtımlarının Karakteristik Özelleri

Linux dağıtımlarının belli başlı karakteristik özellikleri aşağıdaki gibidir.

1. Taşınabilir: Linux işletim sistemi birçok farklı donanım türleri üzerinde çalışabilir ve Linux çekirdeği hemen hemen her türlü donanım platformunun kurulumunu destekler (MacKinnon, 1999).
2. Açık kaynak kod: Linux işletim sisteminin kaynak kodu ücretsiz olarak kullanılabilir ve Linux işletim sisteminin yeteneğini arttırmak için birçok geliştirici işbirliği içinde çalışır (Gonzalez-Barahona vd, 2005).
3. Çok kullanıcılı: Linux işletim sistemi çok kullanıcılı bir sistemdir, yani birden çok kullanıcı Rastgele Erişimli Bellek (RAM), kalıcı bellek veya uygulama programları gibi sistem kaynaklarına aynı anda erişebilir (Krstajić, 2019).
4. Çoklu programlama: Linux işletim sistemi çoklu programlama sistemidir, yani birden fazla uygulama aynı anda çalışabilir (Krstajić, 2019).
5. Hiyerarşik dosya sistemi: Linux işletim sistemi, sistem dosyalarının ve kullanıcı dosyalarının düzenlendiği standart bir dosya yapısı sağlar (Krstajić, 2019).
6. Shell (Kabuk): Linux işletim sistemi, işletim sisteminin komutlarını yürütmek için kullanılabilecek özel bir tercüman programı sunmaktadır. Sistem çağrısı uygulama programları gibi çeşitli işlemler yapmak için kullanılabilir (MacKinnon, 1999).
7. Güvenlik: Linux işletim sistemi, veri şifrelemesi, parola koruması veya belirli dosyalara kontrollü erişim gibi kimlik doğrulama özelliklerini kullanan kullanıcı güvenlik sistemleri sunar (Fenzi & Wreski, 2000).

BÖLÜM 3

DAĞITIMLARDA UYGULAMA PAKETLERİ

3.1. Yazılım Paketleri ve Paket Formatı

Bir paket dosyası genellikle, kurulum komutlarıyla birlikte, derlenmiş ikili dosyalar ve yazılımı oluşturan diğer kaynakları içeren bir dosya arşividir ve uygulamanın tam adı, amacının açıklaması, sürüm numarası, dağıtıcı, sağlama toplamı[1] ve yazılımın düzgün çalışması için gerekli olan bağımlılıkların listesi gibi yazılım ve meta verileri içerir. İşletim sistemlerinde sistemin kararlı bir şekilde çalışabilmesi için farklı görevleri yerine getiren birçok dosya türü bulunur (Mongkolluksame vd, 2012).

İşletim sisteminin dosya türlerini tanıması ve kullanması kolay olabilir ancak sıra kullanıcıya geldiğinde yüzlerce farklı dosya türünü tanıması ve çalıştırması kolay olmaz. Bu yüzden kullanıcının amaçları doğrultusunda kullanacağı uygulamaların kolay kurulumunu yapıp çalıştırabileceği şekilde bir arada ve standart olması gerekmektedir. En azından dağıtımlarda bu uygulamaları kurulum, çalıştırma ve kaldırma işlemlerini yapabileceği uygun bir paket yönetim sisteminin olması gerekmektedir.

3.2. Paket Dönüştürme

Dağıtımların bazıları ayrı bir paket formatı kullanırken bazıları da ortak bir paket formatını kullanır. Genelde hepsinin yaptığı iş ikili dosya türünü çalıştırmaktır.

Paket formatları ikili dosya türünün sıkıştırılmış biçimleridir. Geleneksel uygulama paketleri arasında format dönüşümü yapılabilmektedir. Dönüştürme dağıtım için ihtiyaç duyulan bir paket bulunmadığında faydalı bir yöntemdir. Yine de bu

[1] Sağlama toplamı, iletimi veya saklanması sırasında ortaya çıkmış olabilecek hataları tespit etmek amacıyla bir dijital veri bloğundan türetilen küçük boyutlu bir veridir. Genellikle indirme sunucusundan alındıktan sonra kurulum dosyasına uygulanır. Kendi başlarına, sağlama toplamları genellikle veri bütünlüğünü doğrulamak için kullanılır, ancak veri özgünlüğünü doğrulamak için güvenilmezdir (Sivathanu, Wright & Zadok, 2014).

dönüştürme yöntemi istenmeyen hatalara sebebiyet verebilmektedir. Bu hatalardan kaçınmak için her zaman kendi dağıtımının paket dosyasını bulmaya çalışmak gerekmektedir. Bu dönüştürme uygulamaları hem Grafiksel Kullanıcı Arayüzü (GUI) hem de Komut Satırı Arayüzü'nden (CLI) dönüşüm sağlamaktadırlar. Şekil 3.1'de görsel dönüşüm uygulaması görünmektedir.



Şekil 3.1. Geleneksel Paket Formatları Dönüştürme Uygulaması(Linux Mint, 2019).

Debian ve Red Hat türevi dağıtımlarda CLI'den paket formatlarını dönüştürmek için alien komutundan yararlanılmaktadır (Ambler, 2002). Aşağıda bunlara ait örnekler yer almaktadır.

Debian türevi dağıtımlarda rpm dosya paketini deb dosya paketine dönüştürmek için;

sudo apt update

apt install alien

alien -d dosya.rpm

deb dosya paketini rpm dosya paketine dönüştürmek için;

sudo apt update

apt install alien

alien -r dosya.deb

Red Hat dağıtımlarında deb dosya paketini rpm dosya paketine dönüştürmek için;

sudo dnf update

dnf install alien

alien -r dosya.deb

rpm dosya paketini deb dosya paketine dönüştürmek için aşağıdaki komutlar gerçekleştirilir.

sudo dnf update

dnf install alien

alien -d dosya.rpm

Diğer taraftan terminal ya da CLI'de herhangi bir komutun kullanım kılavuzunu görüntülemek için kullanılan man komutunun alien paket dönüştürme uygulaması ile ilgili uyarısını dikkate almak gerekir. Beklenmeyen sonuçlarla karşılaşmamak ve sistemde kırılmalara yol açmamak için bu uyarı önemlidir. Ubuntu sayfasındaki uyarıda Türkçe şu şekilde denilmektedir (2019),

> (…)UYARI
>
> alien komutu, init, libc gibi önemli sistem paketlerini veya sisteminizin çalışması için gerekli olan diğer şeyleri değiştirmek için kullanılmamalıdır. Bu paketlerin çoğu, farklı dağıtımlar tarafından farklı kurulur ve farklı dağıtımlardan gelen paketler birbirinin yerine kullanılamaz. Genel olarak, sisteminizi bozmadan bir paketi çıkaramazsanız, onu alien komutu ile üretilen bir sürümle değiştirmeye çalışmayın.

Dağıtımlarda varsayılan olarak gelen paket formatı haricinde paket kurmanın bir diğer yolu da paketin kaynak kodundan derleme yapmaktır. Bu şekilde kaynak kodundan istenilen dağıtıma özel kuruluma hazır paket derlemek mümkündür. Kaynak kodları genellikle .tar.gz veya .tar.bz2 uzantılarına sahip sıkıştırılmış tar dosyaları biçiminde bulunurlar.

3.3. Paket Bağımlılığı

Dağıtım çalışma sisteminde kullanıcının kullandığı uygulamaların çalışmasına yardımcı olan kütüphane dosyaları ortak bir klasör içinde tutulur. Kütüphane dosyaları, başka uygulamalara servis sunan, alt uygulamaları içeren yapılardır. Bu şekilde sistemin modüler bir yapısı vardır ve sistemin daha az kaynak tüketmesini sağlarlar. Çoğu dağıtım uygulamasının küçük boyutlarda olmasının nedeni, uygulamanın kullandığı kitaplık dosyalarının ya da bazı araçların kurulum paketinin dışına çıkartılmasıdır. Böylece uygulama, oldukça küçük boyutlara düşebilir. Ancak dağıtımda herhangi bir uygulama paketi tıkla-kur olarak yüklenmek istendiğinde, kurulum için gerekli bazı paketlerin eksik olduğu nedeniyle, kurulum iptal edilebilir. Bu nedenle eksik paketler ya da alt uygulamalar tamamlandıktan sonra, kurulum işlemi gerçekleştirilebilir. Hatta bu sorun yazılım literatürüne İngilizce "Dependency Hell" Türkçe çevirisi de "Bağımlılık Cehennemi" olarak girmiştir.

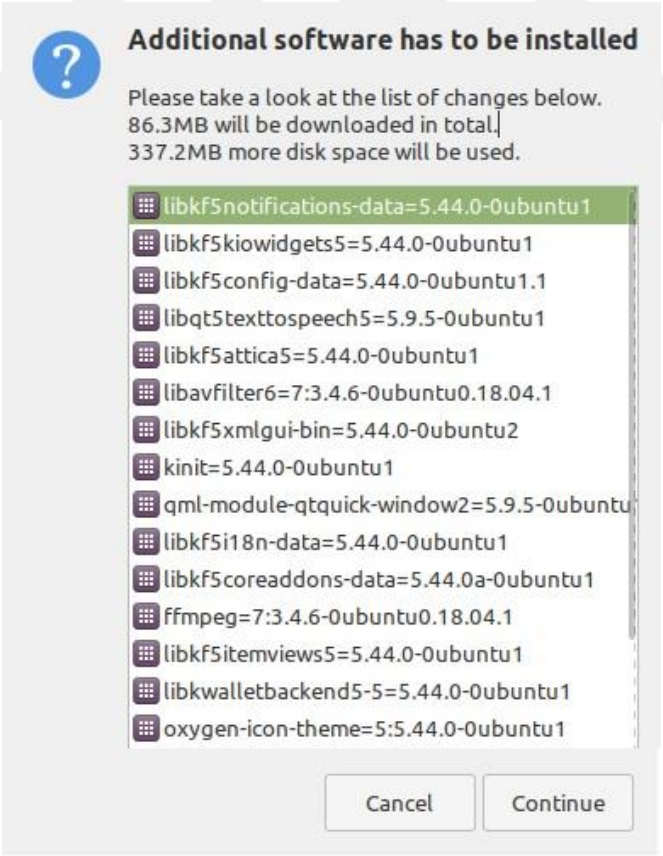
Birçok dağıtımın paket yöneticisi, uygulamaların yükseltme işlemlerinden muzdarip durumdadır. Bir yükseltme yaptığınızda ister tek bir uygulama için isterse işletim sisteminizin tamamı için paket yöneticisi sisteminizde bulunan dosyaların daha yeni sürümlerinin üzerine yazacaktır. Paket yükseltme işlemlerinde herhangi bir sorun olduğunda paket yöneticisi geriye dönük olarak işlemleri yürütemez ise sorunlar kalıcı olacaktır ve uygulama istenilen gibi çalışmayacaktır (Dumitraş, Dig & Neamtiu, 2008).

Firefox İnternet tarayıcı uygulamasının sürümü yükseltildiği zaman paket yönetici de daha yeni bir Çoklu Platform Geliştirme Araç Takımı (GTK) sürümüne ihtiyacı olduğuna karar verir. Yeni GTK geriye dönük uyumlu değilse, sistemdeki diğer uygulamalar aniden kırılabilir. Diğer taraftan bağımlılık sorunları sadece Linux dağıtımlarında meydana gelmez diğer işletim sistemlerinde de bu gibi sorunlar ortaya çıkabilir. Windows işletim sistemi dünyasında da benzer bir sorun Dinamik Link Kütüphanesi (DLL) cehennemi olarak bilinir (Spinellis, 2012). Ayrıca, bu yeni

bağımlılıkların yüklenmesi, yükseltmeyi geri almayı zorlaştırır. Siz veya paket yöneticiniz değiştirilen tüm dosyaların yedeğini almadıkça yükseltme işlemini kolayca geri alamazsınız.

Bir Linux dağıtımı çok sayıda yazılım paketinden oluşur. Paketler arasında çeşitli karmaşık bağımlılık ilişkileri vardır ve bunların yönetimi Linux işletim sisteminin kurulmasının ve sürdürülmesinin temelidir (Lan, Duan, Gao, Zhou & Zhao, 2009).

Bağımlılık sıkıntılarından kurtulmanın en pratik yolu, elle kurulum yapmaktan ziyade mümkün olduğunca çevrimiçi bir şekilde paket yönetim sistemlerini kullanmaktır. Paket yönetim sistemleri, bir uygulama yükleneceği zaman, uygulamanın hangi paketlere ya da alt uygulamalara ihtiyacı olduğunu tespit eder. Eksik olan araçlar ve kütüphane dosyaları tespit edildikten sonra yüklenir. Uygulama Şekil 3.2'de kaşımıza çıkan bütün bağımlılıklar sağlandıktan sonra kendiliğinden kurulur ve bu sayede, hiçbir şey yapmadan rahat bir şekilde uygulamanın kurulumu sağlanır.



Şekil 3.2. Kdenlive Video Düzenleme Uygulamasının Bağımlılık Listesi(Linux Mint, 2019).

Sonuç olarak, bağımlılıkların yerine getirilmesi sayesinde istenen uygulamalar diskten çalıştırılabilir. Aynı bağımlılıklara sahip yeni paketler sistemde daha önce kurulmuş olan bağımlılıklar sayesinde yüklenmesi artık çok daha hızlı olmaktadır ve bu birkaç paketin aynı bileşenleri kullandığı anlamına gelir.

Dağıtımlarda varsayılan paket yöneticilerine nazaran AppImage, Flatpak ve Snappy gibi yeni nesil paket yöneticileri kurulum paketleri içinde gereken her türlü bağımlılık paketi, araç ve kullanılacak bütün kütüphane dosyalarını içerir. Bu da bu paket yöneticileri ile kurulan uygulamaların boyutunu doğal olarak yükseltir. Hem geleneksel olarak varsayılan hem de modern olarak nitelenen bu paket yönetim sistemlerinin kendilerine özgü avantajları ve dezavantajları bulunmaktadır. Bütün bunlar tez çalışmasının altıncı bölümünde incelenecektir.

3.4. Yazılım Paket Depoları

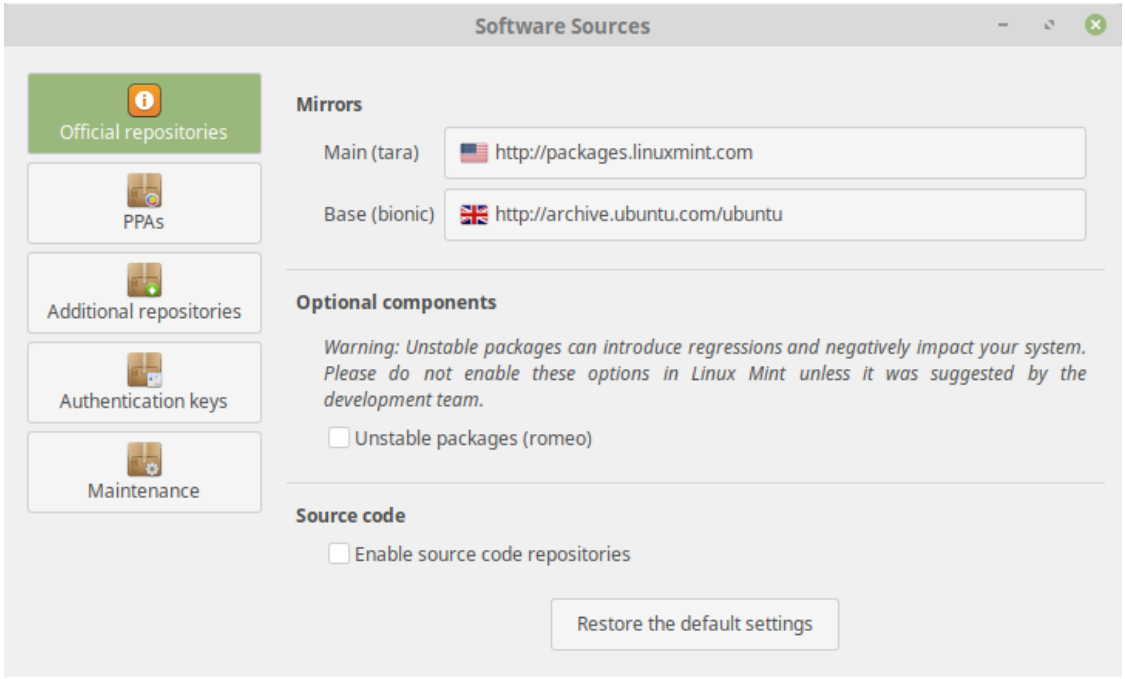
Bir yazılım deposu, kullanıcılar tarafından ücretsiz olarak veya bir ücret karşılığında erişilebilen yazılım veya verilerin saklanabileceği bir yerdir. Dağıtımlarda uygulamaları yükleme, yapılandırma, kaldırma, güncelleme ve paket bağımlılıklarının belirlenmesi ve yüklenmesi gibi işlemlerin gerçekleştirilmesi için paket kaynağı sağlar. Yazılım kaynağı kavramı, kaynak kodunun, okumak veya değiştirmek isteyen herkes için ücretsiz olduğu açık kaynak dünyasında daha önemlidir.

Yazılım depoları, uygulama paketlerini içerir (Koch, 2005). Yazılım merkezi GUI aracılığıyla arama yapıldığında veya CLI'nde sudo apt update veya sudo yum update gibi bir komut kullanıldığında, sisteme bağlı depolardaki tüm paketlerin bir listesi gösterilir. Bir yazılım deposu, dosyalarını bir sunucuda veya yansı (ayna) olarak bilinen birçok farklı sunucuda saklayabilir. Bir dağıtıma ait dünya çapında yüzlerce yazılım paket depo sunucusu olabilir.

Dağıtım, yazılım depolarını sağlayan resmi depolar paketlerin oluşturma yaklaşımlarına göre bazı ayrımlar yaparlar. Örneğin Debian tabanından türetilmiş olan Ubuntu dağıtımı özgür yazılım ile olmayanları, ticari yazılım ile olmayanları ayırmak için farklı depolar sağlamaktadır. Bu kapsamda oluşturduğu dört farklı ana yazılım depoları aşağıda yer almaktadır.

- Ana – Canonical şirketi tarafından destekli ücretsiz ve açık kaynak kodlu yazılımlar.
- Evrensel – Gönüllü topluluk tarafından korunan ücretsiz ve açık kaynak kodlu yazılımlar.
- Kısıtlanmış - Aygıtlar için oluşturulan tescilli sürücü yazılımları.
- Çoklu Evrensel - Telif hakkı veya yasal sorunlarla kısıtlanmış yazılımlar.

Bir bilgisayara herhangi bir dağıtım kurulduğunda o dağıtımın yazılım merkezine bağlı önceden ayarlanmış varsayılan bir depo sunucu adresi ile yapılandırılmış olması gerekmektedir. Örneğin Şekil 3.3'de Linux Mint 19.2 sürümüne ait varsayılan olarak gelen yazılım deposu sunucu adresi görülmektedir. Ancak bazen ihtiyaç duyulan uygulamaların bu depolarda bulunmadığı olabilir. Paket yönetim sistemi bunları GUI ve CLI ile bulamayabilir. Bu gibi durumlarda istenilen uygulamayı bulabilmek için varsayılan depo kaynağına ek olarak kullanıcı tarafından da depolar eklenebilir. Yalnız kullanıcı tarafından elle eklenen bu yazılım depolarının sağlamış olduğu uygulama paketleri varsayılan olanlara göre güvenlik riski taşıyabilir. Çünkü uygulama paketleri varsayılan depolara detaylı bir şekilde test edildikten sonra eklediği için üst düzey güvenlik içerirler. Diğerleri bu kadar güvenli olmayabilir.



Şekil 3.3. Linux Mint 19.2 Dağıtımına Ait Yazılım Deposu Penceresi(Linux Mint, 2019).

3.5. Kişisel Paket Arşivleri

Yazılım Merkezi aracılığıyla arama yapıldığında veya apt-get veya yum gibi bir araç kullanıldığında, sistemde bulunan depolardaki tüm paketlerin bir listesi gösterilir. Bir yazılım deposu, dosyalarını bir sunucuda veya yansı olarak bilinen birçok farklı sunucuda bulundurabilir. Bunlar uygulama paketlerini sağlayan resmi depoları ifade etmektedir. Ancak uygulama mağazaları bir uygulamanın en son sürümüne sahip olmayabilir ve sistemin kararlı çalışması için paketin yeni sürümünün sistemle uyumlu olup olmadığını kontrol etmek için bir dizi kontrolden geçirmek uzunca bir zaman alabilir. Bu sebeple resmi depolar, paketi hemen kullanıcıların hizmetine sunamayacaktır. Aynı şekilde bir geliştirici geliştirdiği uygulamayı resmi depolara eklemek istediğinde resmi depo yöneticilerinin bir karar vermesi ve resmi depolara eklemesi uzun zaman alabilir. Bu gibi problemlerin üstesinden gelmek için dağıtımlar kullanıcılar ya da toplulukların Kişisel Paket Arşivleri (PPA) depolarından paket çekip yüklemesine izin verir.

Birçok geliştirici sadece yazılımın kaynak kodunu sağlar ve uygulamanın kaynak kodundan kurulum yapmak zordur ve bunu herkes yapamaz. Bu nedenle gönüllüler bu kaynak kodundan uygulamayı derleyerek bir PPA depolarına eklemeyi kendileri üstlenirler, böylece diğer kullanıcılar yazılımı kolayca yükleyebilir.

3.6. Yazılım Deposu Kaynak Listesi

Kaynak Listesi, paket yöneticileri ve yazılım depolarını aslında birbirine bağlayan köprüdür. Basitçe ifade etmek gerekirse, kaynak listesi yazılımı barındıran sunucuya Standart Kaynak Bulucu'yu (URL) içeren bir dosyadır. Paket yöneticileri uygulamayı kaynak listeden sağlanan bağlantıdan indirir ve yükleme işlemine devam eder. Bir kullanıcı bu özel PPA depoların bağlantı adreslerini yazılım deposunun kaynak listesi (sources.list)'ne ekleyerek resmi depolardan edinemediği uygulama paketlerini edinebilir. İstenilen kaynak listeleri hem CLI hem de GUI üzerinden eklenebilmektedir.

Kullanıcılar resmi ve resmi olmayan depoların kaynak listelerini sistemine ekleyebilirler. Ancak, hangi depoların hangi uygulamalar için kullanıldığını bilmek

sistem güncel kalması açısından önemlidir. Çünkü güncel uygulamalar sisteme eklenmiş depolardan çekilecektir.

Ubuntu gibi Debian tabanlı sistemde, Kaynak listesi şöyle görünecektir:

deb http://deb.debian.org/debian jessie main

deb-src http://deb.debian.org/debian jessie main

deb http://deb.debian.org/debian jessie-updates main

deb-src http://deb.debian.org/debian jessie-updates main

deb http://security.debian.org/ jessie/updates main

deb-src http://security.debian.org/ jessie/updates main

BÖLÜM 4

PAKET YÖNETİM SİSTEMLERİ

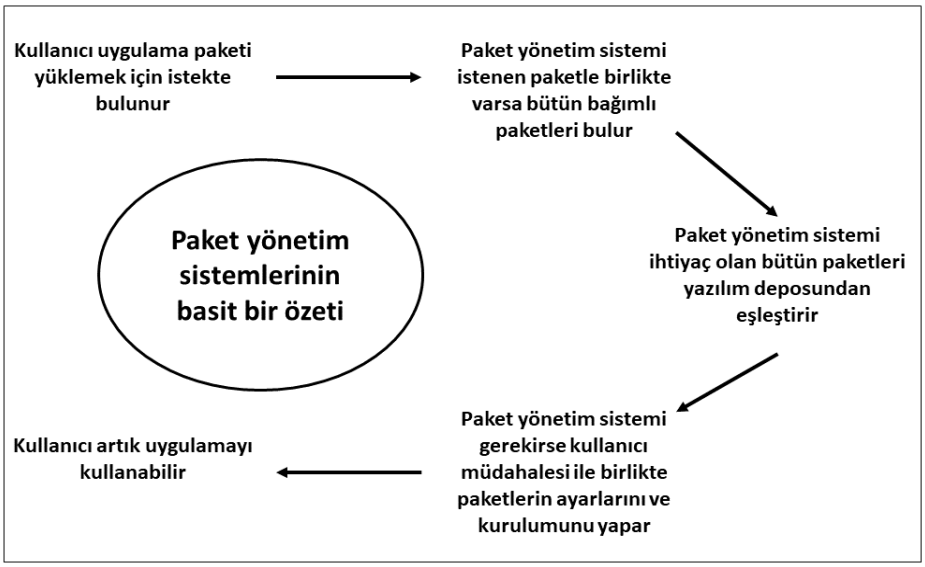
4.1. Paket Yönetim Sistemi

Paket yönetim sistemleri dağıtımlarda uygulamaları yükleme, yapılandırma, kaldırma, güncelleme ve paket bağımlılıklarının belirlenmesi ve yüklenmesi gibi işlemleri kullanıcının ayrı ayrı uğraşmasına gerek kalmadan gerçekleştiren uygulamalardır (Dumitraş, Neamtiu & Tilevich, 2009). Paket yöneticileri, uygulamaların sisteme kurulumunu kolaylaştırır (Jha, 2019).

Temel olarak herhangi bir dağıtımda uygulama yüklemenin beş yöntemi vardır. Bunlar;

1. Konsol(CLI) üzerinden uygulama depolarını kullanarak uygulama yükleme.
2. Masaüstü GUI ile uygulama depolarını kullanarak uygulama yükleme.
3. Sürüm Kontrol Sistemi (CVS) aracılığı ile konsol kullanarak uygulama yükleme. Daha sonra bunun yerini Git platformu aldı.
4. Kaynak kodunu kullanarak uygulama yükleme.
5. Kaynak kodundan hazırlanmış dağıtıma özel uygulama paketleri(deb, rpm. tar.gz, pisi, apk vb.) kullanarak uygulama yükleme.

Bunların arasından GUI ile uygulama yüklemek kullanıcı açısından en kolay olanıdır. Grafik araçları bağımlılık sorunlarının çözülmesine ve yüklemenin doğru şekilde çalıştığının doğrulanmasına yardımcı olur. CLI ve GUI uygulama yüklemek için aynı paket yöneticisini kullanabilirler. İster CLI olsun isterse de GUI olsun bir paketin sisteme kurulma döngüsü Şekil 4.1'deki gibidir.



Şekil 4.1. Bir Uygulamanın Paket Yönetim Sistemindeki Kurulum Adımları.

Dağıtımlara ait CLI ve GUI'de kullanılan bazı paket yönetim sistemi uygulama yükleme araçları aşağıda Çizelge 4.1'de yer almaktadır (Özdemir, 2018).

Çizelge 4.1. Geleneksel Paket Formatları ve Yönetici Araçları.

		Paket Yönetici Araçları	
Linux Dağıtımı	**Paket Formatı**	**CLI**	**GUI**
Debian	deb	apt, dpkg	synaptic
Linux Mint	deb	apt, dpkg	synaptic
Fedora	rpm	Dnf, yum	Dnfdragora, PackageKit
CentOS	rpm	yum	Yum Extender
openSUSE	rpm	zypper	YaST
FreeBSD	txz	make, pkg	OctoPkg
Gentoo	ebuild	emerge, portage	kuroo
Arch Linux	tar	pacman	tkPacman, Pamac
Puppy	pet	PETget	Puppy Package Manager (PPM)
Sabayon	tbz2	enthropy	Rigo
Slackware	txz	slapt-get	XPKGTOOL, sbopkg

4.2. Birincil Paket Yöneticileri

4.2.1. Geleneksel Paket Yönetim Sistemleri ve Formatları

Her işletim sisteminin sistemini kararlı bir şekilde çalıştırmasını sürdürmek için kendine özgü dosya yapısı ve uygulama çalıştırmak için dosya formatı vardır. Dağıtımlar uygulama çalıştırmak için ya kendilerine ait bir paket formatı ya da ortak bir paket formatını kullanırlar. Genellikle bir dağıtımda paket yöneticisi varsayılan paket formatı dışındaki formatların kurulumlarını gerçekleştirmez. Bu yönüyle dağıtımın paket yöneticisiyle kullandığı paket formatı çok yakın ilişkilendirilmiştir.

Oluşturulan bir paket formatını yönetmek için bir paket yöneticisine ihtiyaç duyulmuştur. Birçok paket formatı olmasından dolayı dağıtımlara yönelik uygulama geliştirmek isteyen geliştiriciler için bu durum zorlayıcı bir etken olmuştur. Bunun yanında yapıları bakımından eski birer araç olmaları, bağımlı olunan paketleri içermemesi gibi nedenler geliştiriciler için uygulama paketleme, güncelleme, bakım, bağımlılıkların takibi gibi işlemleri zorlaştırmaktadır.

Bir dağıtımda geleneksel olarak varsayılan paket formatı dışındaki bir formatı kullanmak için paket dönüşümü gerekmektedir. Ancak bu da her zaman tavsiye edilen bir yöntem değildir. Bütün dağıtımlarda paket kurmanın bir diğer yolu da paketin kaynak kodundan derleme yapıp kurmaktır. Paket formatları farklı olsa da hemen hemen hepsi aynı işlevleri yerine getiren arşivlenmiş farklı uzantılara sahip dosyalardır. Aralarında kurulum biçimleri açısından farklar bulunmaktadır.

Paketler geleneksel olarak derlenmiş doğrudan çalıştırılabilir ya da arşivlenmiş kaynak kodları halinde bulunabilirler. Derlenmiş paketler çeşitli uzantılarda olabilirler. Bunlardan Debian dağıtımı tarafından 1995'te üretilen '.deb' ve Red Hat dağıtımı tarafından 1997'de üretilen '.rpm' en yaygın kullanılanlardır. Söz konusu paketlerin yaygın olarak kullanılmasının en önemli sebebi Debian ve Redhat'ın oldukça eski Linux dağıtımları olması ve birçok dağıtımın da bunların türevi olarak ortaya çıkartılmasıdır. Mandriva, Fedora ve CentOS '.rpm' kullanırken, Ubuntu, Linux Mint ve Knoppix '.deb' paketlerini kullanmaktadır. Öte yandan, bazı Linux dağıtımlarının kendi paketleme sistemleri bulunmaktadır. Örneğin Gentoo dağıtımı ebuild paketleme sistemini kullanır. Benzer şekilde, çok yaygın bir Linux dağıtımı olan Slackware de

kendi paketleme formatı olan “.tgz” kullanır (Mancinelli, Boender, Di Cosmo, Vouillon & Durak, 2006).

Arşivlenmiş kaynak kodları yaygın olarak ‘.tar.gz’ ya da ‘.tar.bz2’ formatları ile sunulmaktadırlar. Buradaki ‘.tar’ dosya arşivleme formatıdır. .tar kullanılmasındaki amaç birçok bileşenden oluşan kaynak kodlarını tek bir paket haline getirmek ve arşivdeki dosyaların tarihlerinde bir değişiklik yapmamaktır. ‘.gz’ ve ‘.bz2’ ise ‘gzip’ ve ‘bzip2’ uygulamalarının dosya sıkıştırma formatlarıdır. Kaynak kodlarının büyük boyutlarda olmaması için internete sıkıştırılarak verilirler.

Standart bir dağıtımda her biri kendisiyle bağlantılı özel bir anlamı olan ext2, ext3, ext4, jfs, ReiserFS, XFS ve Btrfs gibi dosya biçimleriyle disk bölümleme seçenekleri vardır(Kára & Lihovarská, 2009). Bu çalışmada karşılaştırılması yapılan modern ve geleneksel uygulama formatları Linux Mint 19.2 ve Fedora 29.1.2 dağıtımında ön tanımlı olarak gelen ext4 dosya sistemi üzerinde incelemeleri yapılmıştır. Geleneksel uygulama formatları paketleri doğrudan ext4 dosya sistemi üzerinde çalıştırırken modern uygulama formatları bu dosya sistemi üzerinde ilave bir katman daha oluşturarak ayrı bir dosya sistemi kullanırlar.

Debian ve Red Hat gibi dağıtımların paket formatları tarafından oluşturulan modüler yapı Linux işletim sisteminin tarihsel gelişimine çok büyük katkıda bulunmuştur. Bununla birlikte, bu yaygın kullanılan formatlar 1990'lı yılların ortalarında İnternetin gelişmekte olduğu dönemde genellikle iş istasyonları ve sunucular için çok rağbet görmesinden dolayı bugünkü bakış açımızdan daha farklı şekillenmiştir. Bugün masaüstü kullanıcıların çoğunun uygulamalarını bir çırpıda bilgisayarlarına kuramamasının nedenlerinin altında bu yatmaktadır.

4.3. Çoklu Paket Yöneticileri

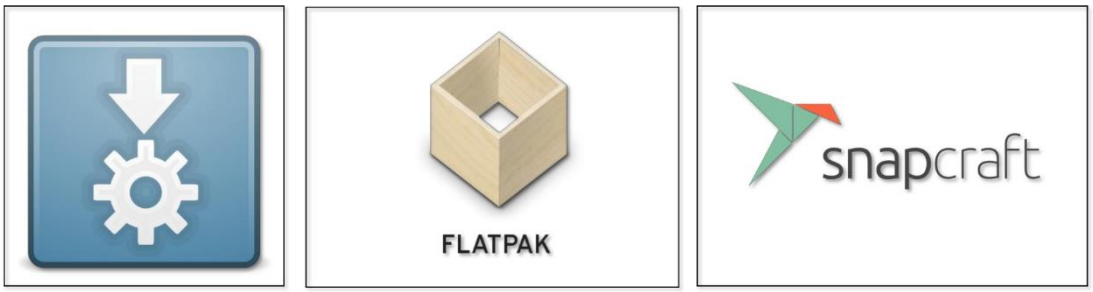
4.3.1. Modern Paket Yönetim Sistemleri ve Formatları

Her dağıtım uygulama çalıştırmak için kendisine ait paket yönetim sistemini kullanır. Ancak dağıtımdan bağımsız bir şekilde uygulamayı tek bir pakette çalıştırmak için gereken tüm bağımlılıklarıyla birlikte o paket yöneticisinin uyumluluğunu destekleyen hemen hemen her dağıtımında çalışabilen paket yöneticileri de vardır. Bu, herhangi bir dağıtımda bir uygulama paketinin bağımlılıklarını gerektiren ve belirli bir

dağıtımda çalışacak şekilde yapılandırılmış olan deb ve rpm gibi geleneksel paket formatlarından farklıdır. Geliştiriciler, belirli dağıtımlarında çalıştırmak için yazılımlarını ayrı ayrı paketlemek yerine, çoğu dağıtımda ortak çalıştırılabilecek tek bir paket oluşturmaktadırlar (Abate, DiCosmo, Treinen & Zacchiroli, 2011).

Dağıtımlarda bir uygulama çalıştırmak, CLI ve GUI vasıtasıyla yazılım merkezini açmak, onu aramak ve yüklemek kadar kolaydır. Uygulama mağazasında bulunmayan uygulamalar, deb veya rpm gibi dosya paketleri üzerinden kurulabilir. Bazılarına PPA'lar üzerinden erişilebilir ve eğer buralardan da bulunmaz ise o zaman uygulamanın kaynak kodundan da derlenebilir. Bunlara rağmen uygulama edinme konusunda yine de bazı kısıtlamalar olabilir. Uygulama mağazaları genellikle bir uygulamanın en son sürümüne sahip olmayabilir, paket bağımlılıkları yorucu olabilir, PPA'lar her zaman güvenli olmayabilir ve kaynak kodundan da uygulamayı derlemek biraz terminal bilgisi ve çalışması gerektirebilir. Çeşitli dağıtımlar ve paket yönetim sistemleri ile hangi dağıtımı kullandığınıza bakılmaksızın bu saydığımız sorunlardan kullanıcıları kurtararak bir uygulamayı çalıştırabilecek modern paket yöneticileri olan AppImage, Flatpak ve Snappy kullanıcıların daha kolay uygulamalara erişiminin sağlanması ve kurulumlarını yapabilmesini amaçlayan yeni nesil paket yöneticileridir ve bunlara ait logolar Şekil 4.2'dedir.

Modern paket yöneticileri kendi içerisinde ayrı bağımlılıklar içerdiğinden, boyut olarak geleneksel emsallerinden daha büyüktürler. Kendi içerisinde bulunan aynı bağımlılık, dağıtımdan bağımsız uygulama paketlerinde ayrı ayrı bulunabilirken, geleneksel paketlerin tümü, yalnızca bir kez sisteme kurulan aynı bağımlılığı kullanırlar.



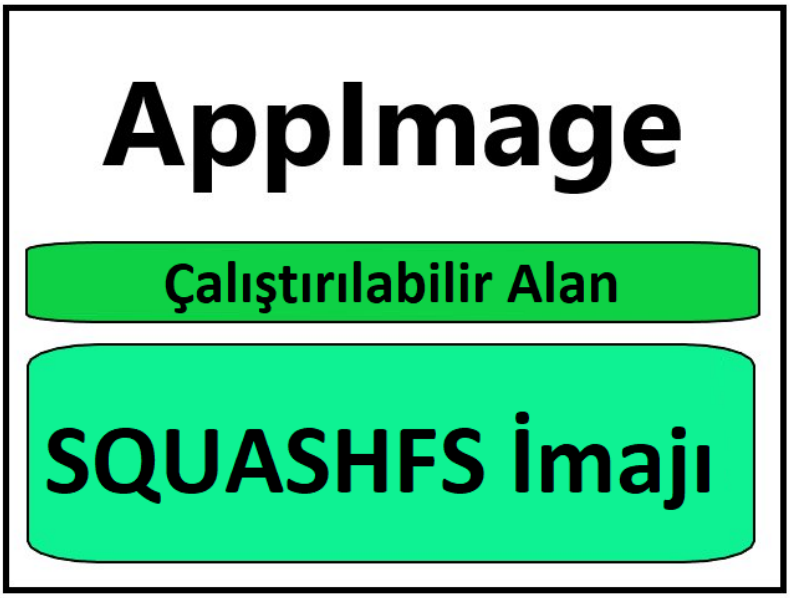
(a) (b) (c)

Şekil 4.2. AppImage(a), Flatpak(b) ve Snappy(c) Paket Yönetim Sistemlerinin Logoları (AppImage, Flathub, Snapcraft, 2019).

Dağıtımlarda paket formatlarını ya da paketleri yönetmek için bir paket yöneticisine ihtiyaç duyulduğu için nasıl ki geleneksel olarak Debian tabanlı dağıtımlarda .deb uzantılı paket formatları ve Red Hat tabanlı dağıtımlarda .rpm uzantılı paket formatları varsa aşağıda açıklanan modern dağıtımdan bağımsız AppImage paket yöneticisinin .appimage, Flatpak paket yöneticisinin .flatpakref ve Snappy paket yöneticisinin .snap uzantılı paket formatları bulunmaktadır. Bu sayılan yeni nesil paket formatlarının her biri kendi web sitelerinde uygulamalarını kullanıcıların hizmetine sunmaktadır. Ayrıca bazı dağıtımlarda yerleşik olarak dağıtımın içinde gelmekle birlikte çoğu dağıtıma da kullanıcı tarafından sonradan yüklenebilmektedir.

4.3.1.1. AppImage

AppImage, ilk olarak 2004 yılında piyasaya sürülen, istenilen yazılımı çalıştırmak için gereken tüm bağımlılıklar ve kütüphaneler ile sıkıştırılmış taşınabilir bir paket biçimidir. Taşınabilir olduğundan, bir AppImage uygulaması çalıştırmak için daha önceden bir kurulum yapılmasına gerek yoktur. Genel olarak bağımlılıklarını içinde barındırması bakımından çalışma prensibi burada incelenen Flatpak ve Snappy paket yöneticilerine benzemektedir. AppImage, özellikle masaüstü ortamlarında çalışmaya odaklanırken sınırlı CLI araçlarıyla sınırlı işlevselliğe sahip sunucularda da çalıştırılabilir.



Şekil 4.3. AppImage Uygulama Yapısı (LinuxCommunity, 2019).

AppImage, çalıştırılabilir uygulama yaptıktan sonra herhangi bir klasör içerisinde hemen kullanılabilen tek bir dosyadır. Daha önce bu uygulamaları çalıştırmak için birinci tipte kullanılan Uluslararası Standartlar Örgütü(ISO) 9660 dosya formatının yerine artık son zamanlarda Şekil 4.3'de de görüldüğü üzere ikinci tip olarak bilinen SQUASHFS[2] dosya sistemiyle kullanılmaya devam etmektedir(AppImage, 2019). Uygulamayı kaldırmak için AppImage dosyasını bulunduğu klasörde silmek yeterlidir. Bu paket formatının kaynak kodu Github web sayfasında mevcuttur. Kullanılabilir paketlerin bir listesi de AppImage web sitesinde bulunmaktadır. Sitede Aralık 2019 itibariyle 946 paket bulunmaktadır (AppImage, 2019). AppImage paketi, güncellemeleri sağlamak için bir depo kullanmaz. Bunun yerine, bir paketi, paketin nasıl güncellenebileceği hakkında bilgiler içerebilir. Bu güncelleme bilgilerini içeren AppImage için, AppImageUpdater gibi ayrı bir araç paketleri el ile güncellenmesine yardımcı olabilir.

4.3.1.2. Flatpak

Flatpak, İsveçli geliştirici olan Alexander Larsson tarafından 2007'de bir proje olarak ortaya çıkmıştır. Flatpak paket yöneticisi, 2015 yılında ilk kez piyasaya açık kaynak kodlu olarak sürülmüştür. Larsson, Flatpak projesinin geliştirme aşamasında Red Hat işletim sisteminin birçok bileşenini kullandığı için ona atfedilerek geliştirilmiştir (Höbel, 2018). Bu paket yöneticisi de aynı Snappy paketleri gibi uygulamaları sistemin geri kalanından izole etmek için tasarlanmıştır. Flatpak kaynak kodu Github web sayfasında bulunmaktadır.

Flathub web sitesi, Flatpak paketlerini bulmak ve kurmak için kullanılıyor. Flatpak paket biçiminin açık kaynak kodlu veya özel yazılımlarla ilgili herhangi bir kısıtlaması olmamasına rağmen, Flathub uygulama mağazası öncelikli olarak açık kaynak kodlu yazılımların web sitelerinde yayınlanmasına izin veriyor. Son zamanlarda bazı özel yazılımlar da uygulama mağazasına girmiştir. Snappy paket yöneticisinin Canonical tarafından kontrol edilen tek deposunun aksine, Flatpak, paketlerini yüklemek ve güncellemek için birden fazla depo kullanılmasına izin verir. Flatpak

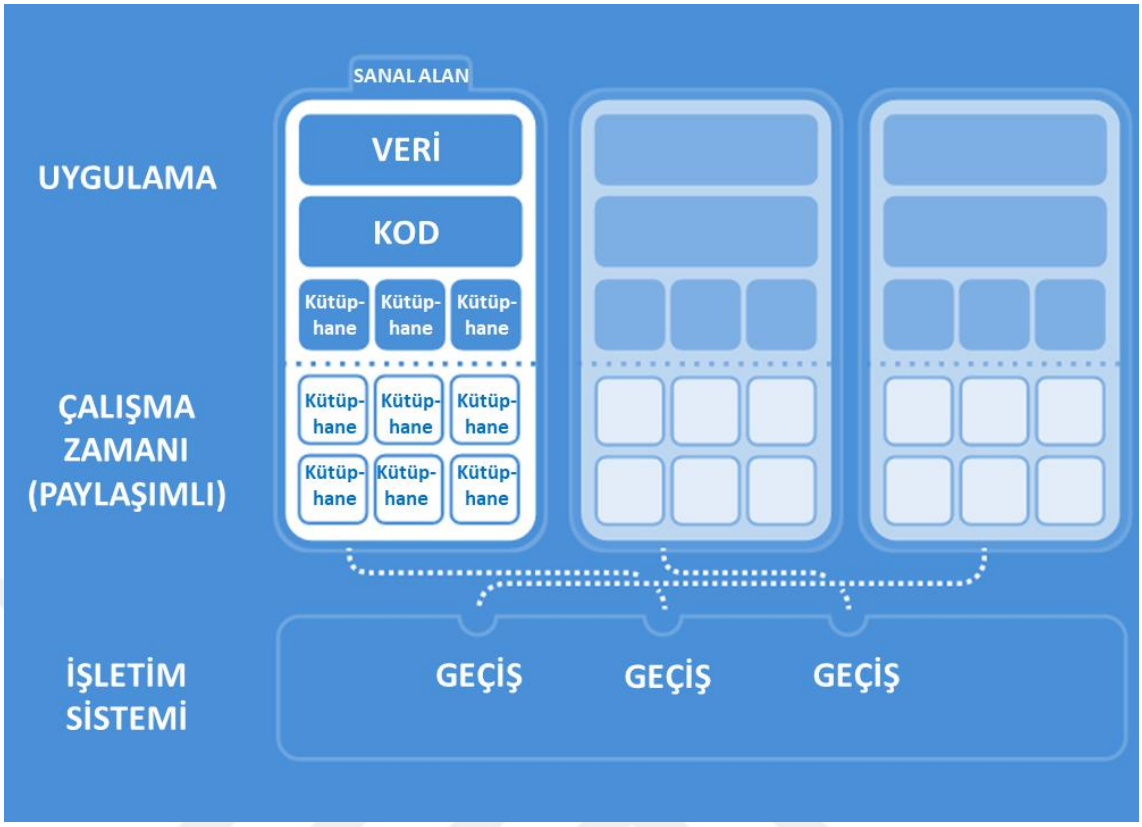
[2] Squashfs, Linux için sıkıştırılmış salt okunur bir dosya sistemidir. Squashfs dosyaları, düğümleri ve dizinleri sıkıştırır. Gömülü sistemler gibi düşük yüke ihtiyaç duyulan alanlarda kullanılmak üzere tasarlanmıştır(Pavlov & Cecchetti, 2008).

uyumluluğu hakkında genel bilgiler ve Flatpak paketlerinin nasıl dağıtılacağı ile ilgili kılavuzlar sitesinde bulunmaktadır (Flathub, 2019).

Flathub web sitesinde Aralık 2019 itibariyle 707 adet Flatpak paketi mevcut, ancak Flatpak'ın tüm paketleri içeren tek bir deposu olmadığından, kullanılabilir toplam Flatpak paketlerinin miktarı muhtemelen daha fazladır (Armasu, 2016).

Flatpak uygulamalarına göz atmayı, kurmayı ve kullanmayı kolaylaştıran GUI'ne sahip yazılım merkezi bulunmaktadır. Genel olarak dağıtımların ileri sürümleriyle uyumludur, yani aynı Flatpak uygulaması, bir dağıtımın sonraki sürümlerinde değişiklik yapmadan çalışabilir. Bununla birlikte, bu paket yöneticisinde oluşturulan uygulamalar genellikle diğer formatlardaki açılışlara göre daha yavaştır ve bazen kullanıcının tercih ettiği sistem temasıyla uyumlu olarak çalışmayabilir. Ayrıca Flatpak paket yöneticisinin büyük bir dezavantajı, Snappy ve AppImage yöneticilerinin aksine tam bir sunucu desteğinin olmamasıdır. Çoğu Linux sunucusunda bulunmayan bazı masaüstü oturumu hizmetlerine bağımlıdır (Flathub, 2019).

Flatpak formatı tüm bağımlılıklarıyla tek bir paket olarak paketlenir. Ancak, bir Flatpak uygulaması içerisinde, sistemdeki birden fazla uygulama arasında paylaşılan ve Şekil 4.4'de de gösterilen bir "çalışma zamanı" ve "kitaplık" koleksiyonu da vardır; bu aynı zamanda bunun burada açıklanan diğer modern paket yöneticilerinden daha az disk alanı kullanması anlamına gelmektedir. Bu uygulama paketinin yöneticisi uygulama çalıştırmak için varsayılan dosya sisteminin üzerinde git benzeri bir modeli önyüklemeli dosya sistemi ağaçlarını işlemek ve indirmek ile bunları dağıtmak ve önyükleyici yapılandırmasını yönetmek için bir katmanla birlikte bir araya getiren bir proje olan OSTree veya libostree katmanını kullanır. OSTree, bir uygulamanın birden çok sürümünün kurulumu için sürüm kontrolü ve paket sistemlerindeki özellikleri birleştirir. Böylece bu, kullanıcıların kaynak kodu deposunda bir değişiklik yapıldıktan sonra çok geçmeden GNOME gibi bir arayüzde uygulamanın derlenmiş biçimlerini denemelerine olanak tanır(Walters, Poo-Caamaño & German, 2013).



Şekil 4.4. Flatpak Uygulama Yapısı (LinuxCommunity, 2019).

4.3.1.3. Winepak

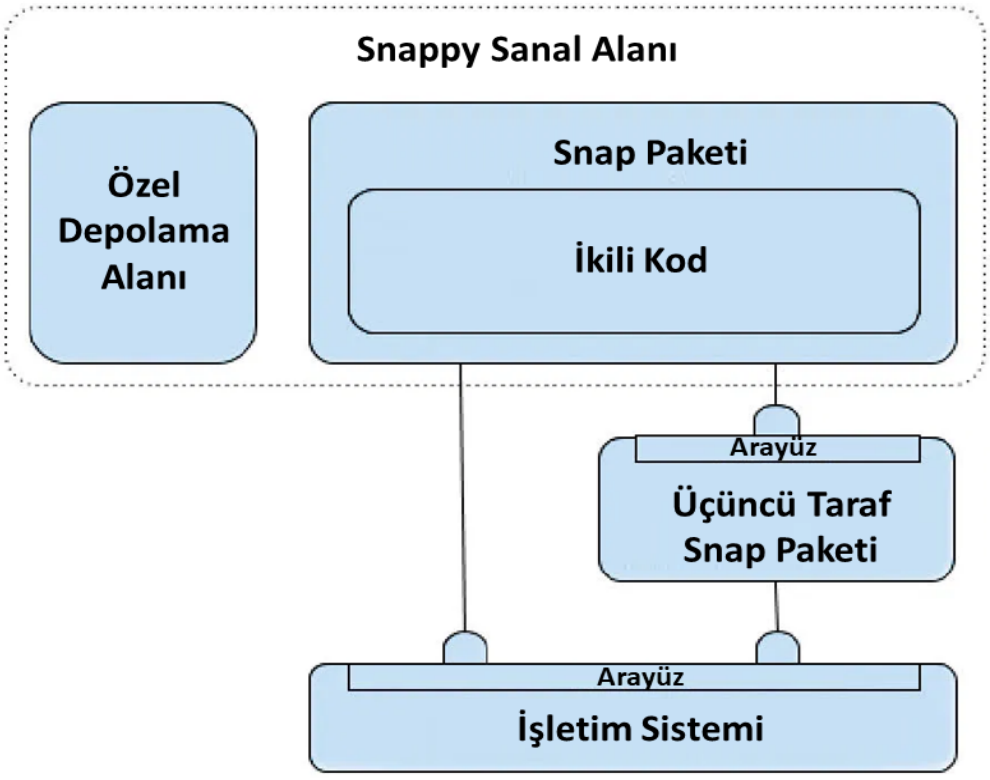
Dağıtımlarda kullanıcılar Microsoft Windows işletim sistemine ait bazı uygulamaları kullanmak istediklerinde zaten Wine Öykünücü Değildir(WINE) isimli uygulama ile kurulumlarını yapabiliyorlardı. Ancak bu uygulamalar bazı uyumsuzlukları nedeniyle kararlı çalışmıyordu. Microsoft Windows uygulamalarını WINE ile kullanabilmenizi sağlayan Winepak ücretsiz, açık kaynak kodlu ve kurulumu kolay bir Flatpak paket deposudur (Logix, 2018). Winepak paket yöneticisinin Github sayfası üzerinde sahip olduğu uygulamalar ve kurulum için gerekli dokümanlar mevcuttur (Winepak, 2019). Winepak kullanabilmek için öncelikle dağıtıma Flatpak paket yöneticisini daha sonra da Winepak kurulumu yapılması gerekmektedir (Wallen, 2018).

4.3.1.4. Snappy

Ubuntu dağıtımını yöneten ve finanse eden Canonical Ltd. adlı özel bir şirket tarafından oluşturulan ve 2014 yılında ilk kez piyasaya sürülen Snappy, dağıtımdan

bağımsız bir paket yönetim sistemidir (Canonical, 2016). Bu paket yöneticisi 2013 yılında Ubuntu'nun telefonunda kullanılan .click paket formatının geliştirilmiş bir versiyonudur. Daha sonra şirket Nesnelerin İnterneti (IoT) platformu için uygulamalar geliştirdi. Ubuntu 16.04 ile birlikte masaüstü ortamına geçiş yaptı. Canonical, Haziran 2016'da, diğer dağıtımlar ve amaçlar için paket formatını kullanıma açtı. Snappy, o zamandan beri büyük dağıtımlar da dâhil olmak üzere şu anda 41 dağıtımla uyumluluğu desteklemektedir. Ayrıca çevrimiçi uygulama mağazası Snapcraft, binlerce Snappy paketlerini kullanıcılarının hizmetine sunmaktadır (Ljubuncic, 2019).

Snapcraft platformunda geliştiriciler, yazılımlarını Snappy paketleri olarak yayınlamalarına yardımcı olacak kılavuzlar bulabilirler. Açık kaynak kodlu ve özel yazılımların her ikisi de Snapcraft uygulama mağazasında yayınlanabilir. Bununla birlikte, Snappy Flatpak yöneticisinin aksine, paketin içinde yer alan tüm "bağımlılıklar", "çalışma zamanları" ve "kitaplıklar" ile birlikte gelir ve bu da uygulamanın boyutunu artırır böylece daha fazla disk alanı tüketir. Şekil 4.5'teki gibi bir Snappy paketi pratik olarak belirli bir uygulamaya adanmış sanal bir makine gibidir. AppImage gibi SQUASHFS dosya sistemi üzerinde çalışır.



Şekil 4.5. Snappy Uygulama Yapısı (LinuxCommunity, 2019).

Snappy üzerinde çalışan uygulamaların kullanıcıları, güvenlik sorunlarından korkmadan kullanabilmeleri için önce paketin onayı sağlanmalıdır. Bununla birlikte Snappy paketleri dağıtıma kurulduktan sonra kullanıcının uygulamayı ilk kez açtığında önemli bir yavaş başlangıç alacağına dikkat edilmelidir, ancak bu yalnızca ilk açtığında olacaktır, daha sonra uygulama neredeyse sıradan bir Debian paketi kadar hızlı çalışacaktır.

Snappy son zamanlarda çok popüler bir paket yöneticisi haline gelmiştir, ancak Canonical şirketinin bu paket yöneticisini ne kadar destekleyeceği tam olarak bilinmiyor. Snapcraft merkezi uygulama mağazası, Snappy paketleri için hem avantaj hem de dezavantaj olarak görülebilir. Kullanıcının tüm mevcut Snappy paketlerini bulabileceği tek bir konuma sahip olması güzel olabilir, ancak bu uygulama mağazası yalnızca Canonical şirketi tarafından kontrol ediliyor ve korunuyor. Bu durum ise özgür yazılım kurallarıyla uyuşmamaktadır.

4.3.2. Kapsayıcı Paket Yöneticileri

4.3.2.1. Sysget

Sysget, neredeyse her dağıtıma özel geleneksel paket yöneticisinin yaptığı görevleri üstenen kapsayıcı bir paket yöneticisidir. Sysget, ücretsiz ve açık kaynak kodlu olarak C++ programlama dili ile oluşturulmuştur. Sysget'in Github web sayfasında desteklediği dağıtımlar ve gerekli dokümanlar bulunmaktadır (Sysget, 2019). Bu paket yöneticisi paketleri aramak, yüklemek, güncellemek ve kaldırmak ile paketlerin veri tabanını güncellemek ve paket yöneticisi önbelleğini temizlemek gibi her türlü temel paket yönetim işlemlerini gerçekleştirebilir. Özellikle bu paket yöneticisi, işlerini komut satırı üzerinden gerçekleştiren ve sık dağıtım değiştiren kullanıcılar için yararlıdır. Sysget, kullanmakta olduğunuz dağıtımın varsayılan paket yöneticisini kullanımdan kaldırmaz sadece kendine özel komut dizimi ile paketler üzerinde işlem yapmaya yardımcı olur.

4.3.2.1. Nix

Hollanda'daki Utrecht Üniversitesi'nde akademik bir proje olarak başlayan yeni nesil bir paket yöneticisi olan Nix, Ubuntu, Debian, SUSE, Fedora ve Red Hat gibi

Linux'un farklı dağıtımlarına dağıtımın varsayılan paket yöneticisine müdahale etmeden ikili ya da kaynak kodlu paket kurmak için dağıtımdan bağımsız bir yöntem sunar (Nix, 2019).

Mevcut paket yöneticilerinden farklı olarak, Nix farklı uygulama sürümlerinin yan yana yüklenmesine izin verir ve uygulama güncellemelerinin akıllıca geri alınmasına izin verir. Nix, farklı kütüphaneler, derleyiciler veya çeviriciler üzerinde desteklenen uygulamalar geliştiren farklı yapıda ortamlar ve geliştiriciler için kullanışlı bir paket yönetim aracıdır. Hatta bu paket yöneticisi daha da ilerletilerek NixOS adında bir GNU / Linux dağıtımı olarak ortaya çıkmıştır (Dolstra, Löh & Pierron, 2008).

Dağıtım tarafından sağlanmayan yeni bir yazılımı dağıtmak gerektiğinde, Nix birden fazla sürüm, tam bağımlılıklar ve geri alma için destek sağlar. Nix, yazılım yönetimi için öngörülebilir bir sistem oluşturarak kullanıcıları bağımlılık cehenneminden kurtarır.

4.3.2.1. GNU Guix

Linux çekirdeğinin değiştirilmiş sürümlerinden sağlanan bir GNU paketi ve işletim sistemi çekirdeği olan Linux-libre çekirdeğinden inşa edilen ve geliştirme aşamasında GNU Hurd çekirdeğini kullanan GNU Guix paket yöneticisi 3 Şubat 2015'te Özgür Yazılım Vakfı'nın özgür dağıtımlar listesine girmiştir. GNU Guix, Scheme[3] programlama dilinin bir çeşidi olan GNU Genişletmeler için Her Yerde Akıllı Dil (GUILE) kullanılarak Nix paket yöneticisinden yeniden üretilmiş, bütünüyle fonksiyonel paket ve yapılandırma yöneticisidir. Bu paket yöneticisi sisteme kurduğu uygulama paketlerini kendi dizinindeki bir paket deposunda bulundurur. Dağıtımlarda dağıtımın varsayılan paket yöneticisine müdahale etmeden paket kurmak için Nix gibi dağıtımdan bağımsız bir yöntem sunar (GNU Guix, 2019).

Uygulama paketlerini yükleme, yapılandırma, kaldırma, güncelleme ve paket bağımlılıklarının belirlenmesi ve yüklenmesi gibi işlemleri yerine getiren paket yöneticisi bu işlemleri paketlenmiş imaj dosyası gibi saklar. Böylece, kullanıcı bu

[3] Scheme, Yüksek Düzey Programlama Dili(AlGOL) ve Konum Belirleyici / Tanımlayıcı Ayırma Protokolü(LISP'ten türetilen genel amaçlı bir programlama dilidir. Kavramsal olarak temiz ve basit olduğu için öğrenmesi kolay bir dildir. Aynı zamanda, güçlü bir dildir ve gücünün tam ustalığı dikkatli bir çalışma ve uygulama gerektirir(Dybvig, 1996).

işlemler sırasında herhangi bir sorunla karşılaşırsa bir önceki durumuna geri dönebilir. Diğer taraftan paket dizinlerinde biriken bu işlem imajları paket yöneticisinin vermiş olduğu imkânlarla istenmeyen işlemlerin eski sürümleri kullanıcılar tarafından silinebilir (Courtès, 2013).

4.3.2.4. Nix ve GNU Guix Paket Yöneticilerinin Karşılaştırılması

Nix ve GNU Guix paket yöneticileri belli başlı özellikleri Çizelge 4.2'de karşılaştırılmıştır (Parrella, 2019).

Çizelge 4.2. Nix ve GNU Guix Paket Yöneticilerinin Karşılaştırılması.

	Nix(OS)	**Guix(SD)**
Paketleme Yazılım Dili	Nix, Bash	Guile Scheme
Uygulama Yazılım Dili	C++	Guile Scheme
Özgür Yazılım	Opsiyonel	Tam
Desteklenen İşletim Sistemleri	Resmi: Linux, Darwin (MacOS) Bazen: Cygwin, BSD, Android, iPhone, RaspPi, Solaris NixOS Linux çekirdeğini kullanır.	Linux, GNU Hurd GuixSD Linux-libre'den oluşturuldu.
Sistem Daemon	systemd	GNU Shephard
Lisanslama	MIT/X11	GPLv3
Depolama Konumu	/nix	/gnu

BÖLÜM 5

UYGULAMALARIN SANAL ALANDA ÇALIŞMASI

Günümüzün bilişim dünyasında İnternet üzerinden gelen kötü amaçlı ya da hatalı içerikler kullandığımız işletim sistemlerine zarar vermektedir. İşletim sistemleri bu olumsuz etkilerden korunmak için çeşitli güvenlik önlemleri alır. Buna rağmen kullanıcılar beklenmeyen olumsuz etkilerden zarar görmemek için ek güvenlik tedbirleri alması gerekmektedir. Aynı şekilde bağımlılık sorunlarından dolayı sistemde uygulama hataları meydana gelmesinden dolayı kullanıcı zarar görebilir. İşte bu noktada sanal alan oluşturma, uygulamaların yalıtılmış ortamda çalışmasını sağlayan önemli bir güvenlik ve koruma yöntemidir(Borate & Chavan, 2016).

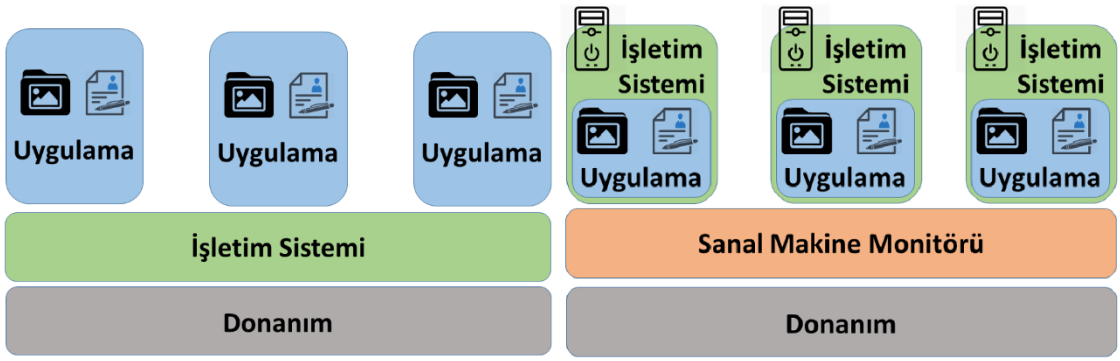
Geleneksel formatlar bağımlılık sistemiyle çalışırlar. Neredeyse bütün sistem birbirine bağlıdır. Herhangi bir yerde kırılma olması bütün sistemi etkileyebilir. Ancak modern formatlar dağıtımların ext4 gibi varsayılan dosya sistemlerinin üzerinde ayrı bir dosya sistemleri oluşturarak tıpkı sanal alanda gibi çalışmaktadırlar.

Uygulama paketlerini bağımlılık sorunlarından kurtarmak için modern paket formatları kullanılabilir. Bir diğer yolu ise bunları sanallaştırılmış alanlarda çalıştırmaktır.

5.1. Sanallaştırma

Bir bilgisayar sisteminin genel olarak donanım, işletim sistemi, sistem yazılımları ve uygulama yazılımları gibi temel bileşenleri olduğundan bahsedildi. Sanallaştırma bilgisayarın işlemci, bellek ve hafıza gibi donanım kaynaklarını paylaşan üzerinde bulunduğu işletim sisteminden ayrı farklı işletim sistemleri çalıştırmaya yarayan donanım katmanı ile kullanıcı arasında Hypervisor veya Sanal Makine Monitörü (VMM) olarak adlandırılan ve Şekil 5.1'de gösterilen mantıksal bir

katmandır. Bu katman fiziksel kaynakların mantıksal kaynaklara dönüştürülmesini sağlar.



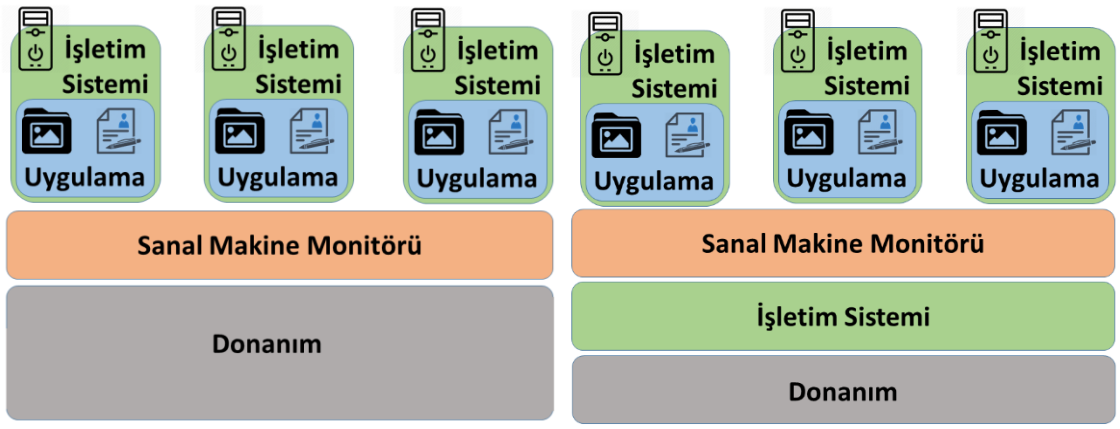
(a) (b)

Şekil 5.1. Sanallaştırma Yapmadan Önceki(a) ve Yapıldıktan Sonraki(b) Sistem Yapısı.

İki tip hypervisor teknolojisi bulunmaktadır (Blokdijk, 2012).

Tip 1 hypervisor: Doğrudan sistem donanımının üzerinde çalışır. Native, Bare metal ve embedded çeşitleri bulunmaktadır.

Tip 2 hypervisor: Şekil 5.2'deki gibi işlemci, bellek ve hafıza yönetimi gibi sanallaştırma hizmetleri sunan bir ana bilgisayar işletim sistemi üzerinde çalışır.



(a) (b)

Şekil 5.2. Tip 1(a) ve Tip 2(b) Hypervisor Yapıları.

Günümüzde Kullanılan Tip 1 ve Tip 2 hypervisor çeşitleri aşağıda sıralanmıştır.

Tip 1 Hypervisor

1. VMware ESX ve ESXi

2. Microsoft Hyper-V

3. Citrix XenServer

4. Oracle VM

5. KVM

Tip 2 Hypervisor

1. VMware İş İstasyonu / Fusion / Oynatıcı

2. VMware Server

3. Microsoft Sanal Bilgisayar

4. Oracle VM VirtualBox

5. Red Hat Kurumsal Sanallaştırma

Sanallaştırma sistemleri bilgi teknolojileri özellikle de bulut bilişim alanında çalışanların kurulum, bakım ve yönetim maliyetlerini azaltırken aynı zamanda yüksek verimlilik, esneklik, sürdürülebilirlik ve daha fazla yatırım imkânı sağlar.

Sanallaştırma kısaca belli bir fiziksel kaynaklar üzerine ya da bu kaynakları kullanan bir işletim sistemi üzerine birçok işletim sistemi kurmaktır. Genel olarak türleri aşağıda yer almaktadır.

- Masaüstü Sanallaştırma
- Uygulama Sanallaştırması
- Sunucu Sanallaştırma
- Depolama Sanallaştırma
- Ağ Sanallaştırması

Sanallaştırma sistemleri bilgi teknolojileri özellikle de bulut bilişim alanında çalışanların kurulum, bakım ve yönetim maliyetlerini azaltırken aynı zamanda yüksek verimlilik, esneklik, sürdürülebilirlik ve daha fazla yatırım imkânı sağlar.

Sanallaştırmanın belli başlı faydaları aşağıda maddeler halinde verilmiştir.

1. Kolay yönetilebilirlik
2. Uyum Problemlerinin Ortadan Kalkması
3. Hata Tespiti ve İzolasyonu
4. Arttırılmış Güvenlik

5. Kaynakları Verimli Kullanma
6. Verilerin Taşınabilirliği
7. Sorunsuz Testler
8. Hızlı Dağıtım
9. Maliyetlerin Azalması
10. Uygulamaları Ayırmak

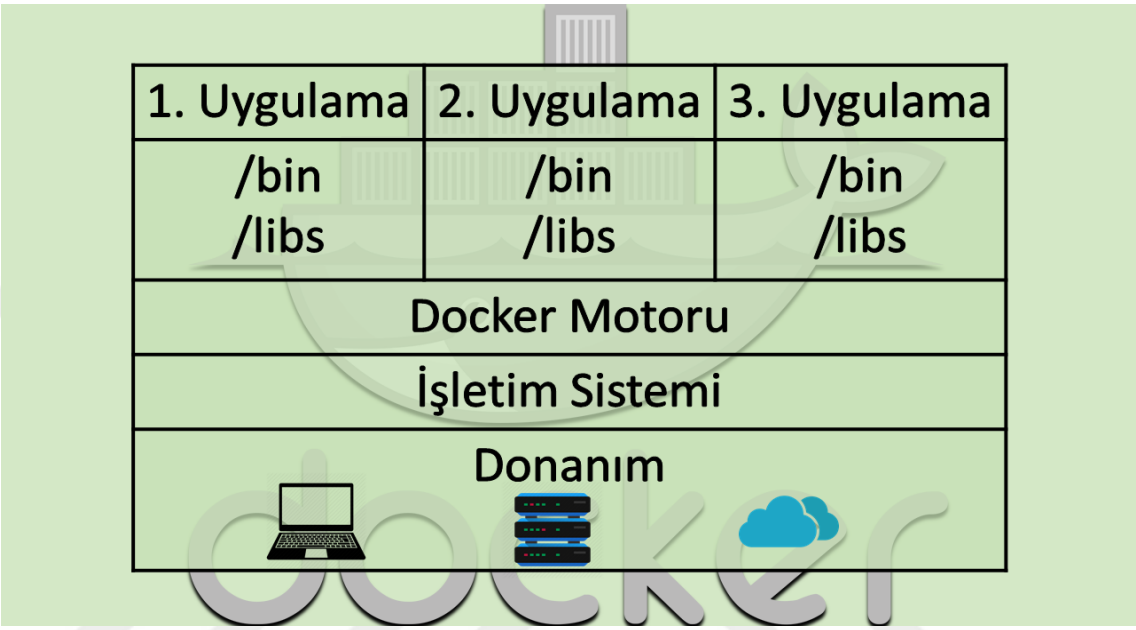
5.2. Konteyner

Bazı uygulamalar farklı sebeplerden dolayı varsayılan işletim sistemi üzerinde çalıştırılmak istenmez. Bunlar ya başka bir donanım üzerindeki işletim sisteminde ya da kendi donanım üzerindeki işletim sistemine kurulan sanal makinalar üzerinde uygulamalar olarak çalıştırılabilir. Sanal makinaların aksine Konteynerler bir uygulamayı çalıştırmak için donanımın kaynaklarını ayrı bir platform üzerinde kullanmaktansa aynı işletim sistemi üzerinde konteyner motoru sayesinde uygulamaları çalıştıran yapılardır (Li, Kihl, Lu & Andersson, 2017). Bu yapıda geliştiricinin oluşturduğu uygulama, uygulamanın tüm bağımlılıkları ve çalışması için gerekli olan diğer paketleri bir konteyner imajı halinde paketlenir.



Şekil 5.3. Bir Donanım Sistemi Üzerindeki Konteyner Yapısı.

Şekil 5.3’de de verildiği gibi konteynerler, bir konteyner motoru tarafından çalıştırılan imajlardır. Uygulamalar, işletim sisteminin diğer işlemlerinden ayrı tamamen izole edilmiş (sandbox) bir alan olan konteyner içinde çalışır. İşletim sistemi çekirdeğinde doğrudan desteklenir ve bu sayede daha az donanım kaynağı tüketerek, daha verimli ve daha hızlı çalışır (Syed & Fernandez, 2015).



Şekil 5.4. Docker Konteyner Uygulamalarının Bir Sistemdeki Yapısı.

Günümüzde birçok konteyner teknolojisi olmasına rağmen en yaygın olarak kullanılan konteynerleştirme platformu, yapısı Şekil 5.4’de de verilen Docker konteyneridir. Konteynerleştirme kısaca konteyner gibi bir kapsayıcının içine paketleme anlamına gelmektedir. Aynı işletim sistemi içerisindeki konteynerler çok zaruri olmadıkça birbiri ile iletişime geçmezler. Bunun amacı güvenlik açısından konteynerleri korumaktır.

2008 yılında Linux çekirdeğine eklenen Linux Konteynerleri(LXC) üzerine kurulu olan Docker konteyner teknolojisi bir işletim sistemi içerisinde birbirinden yalıtılmış bir şekilde çalışan konteynerler bir sanal alan sağlamaktadır. LXC’nin Docker kadar popüler olamamasının temelinde hem kurulum zorlukları hem de sonrasında yapılandırma sıkıntıları yatmaktadır.

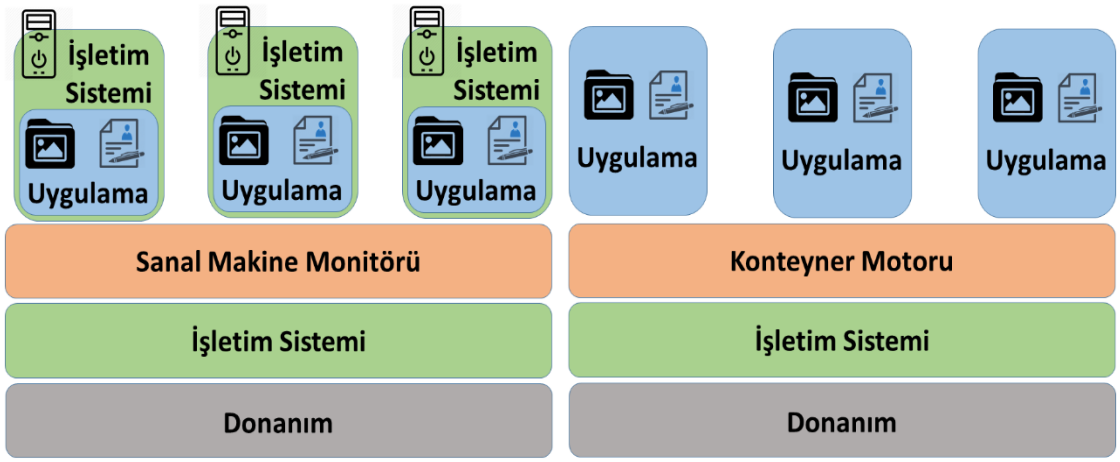
Docker LXC yapısını basitleştirerek uygulamaları konteyner olarak sunarken Ubuntu dağıtımının ana destekleyicisi Güney Afrika'lı girişimci Mark Shuttleworth'e ait

Canonical Ltd. tarafından LXC yapısından esinlenerek oluşturulan Linux Konteynerleri(LXD) VMM gibi davranmaktadır (Kumaran, 2017). Bunun yanı sıra rkt, konteyner motoru da CoreOS tarafından geliştirilen ve açık kaynak kodlu bir konteyner sistemidir (CoreOS, 2019).

5.3. Sanal Makine Monitörü ve Konteyner Arasındaki Farklar

Sanal makine monitörü ve konteynerler bilişim teknolojileri alanında yapılan hizmetlerin kurulum, bakım ve yönetim maliyetlerini azaltırken aynı zamanda yüksek verimlilik, esneklik ve sürdürülebilirlik imkânı sağlaması bakımından birden çok izole edilmiş hizmeti tek bir platformda çalıştırmanın iki önemli sanallaştırma yöntemidir.

Sanal makine monitörleri genellikle bir işletim sistemi üzerinde bir katman olarak çalışmak zorunda olduğundan üzerinde bulunduğu donanımın kaynaklarını önemli ölçüde tüketirler. Bunlara kıyasla konteynerlerin doğrudan işletim sistemiyle birlikte çalışması onu daha avantajlı hale getirmiştir. Şekil 5.5'de de uygulamaların donanıma olan yakınları görülmektedir.



(a) (b)

Şekil 5.5. Sanal Makine Monitörü(a) ve Konteyner(b) Arasındaki Yapı Farkı.

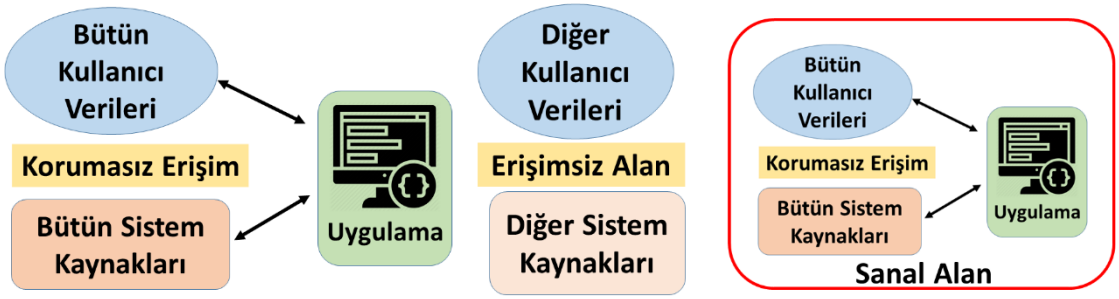
5.4. İşletim Sistemi İmajı

İşletim sistemi imajı kullanılan herhangi bir işletim sisteminin bir veya daha fazla bölümlerinin işletim sisteminin varsayılan uygulamasıyla ya da bunların haricindeki uygulamalarla bir bütün olarak dosya haline getirilmesidir. İmaj alındıktan

sonra aynı bilgisayara ya da aynı donanıma sahip başka bir bilgisayara geri yükleme yapılarak birebir olarak imaj alındığı gibi işletim sistemi kullanılabilir. Bazı durumlarda aynı donanıma sahip olamayan bilgisayarlara da desteklemesi halinde kurulum yapılabilmektedir.

5.5. Sanal Alan

Bilinmeyen kaynaklardan gelen uygulamaların çalıştırılması ya da resim, video, sıkıştırılmış dosya vb. dosyaların bilgisayara indirilmesi potansiyel olarak bilgisayara zararlı yazılımlar bulaştırmasına ve kötü niyetli bilgisayar korsanlarının kullanıcı verilerine erişmesine sebep olabilir (Viswanathan & Neuman, 2013). Sonuçlarının tam olarak nelere sebebiyet vereceği bilinmeyen ya da bazı şüpheli uygulamaları işletim sistemini etkilemeden güvenle çalıştırmak için sanal alan (sandboxing) yapılarına ihtiyaç duyulmuştur. Sanal alan Linux çekirdeğinin desteği ile uygulamayı sistemin geri kalanından ayırarak çalışan uygulamaya korumalı bir alan sunar (Laurén, Rauti & Leppänen, 2017). Şekil 5.6'da sanal alanda olan ve olamayan bir uygulamanın bütün kullanıcı verilerine ve sistem kaynaklarına erişimleri görünmektedir.



(a) (b)

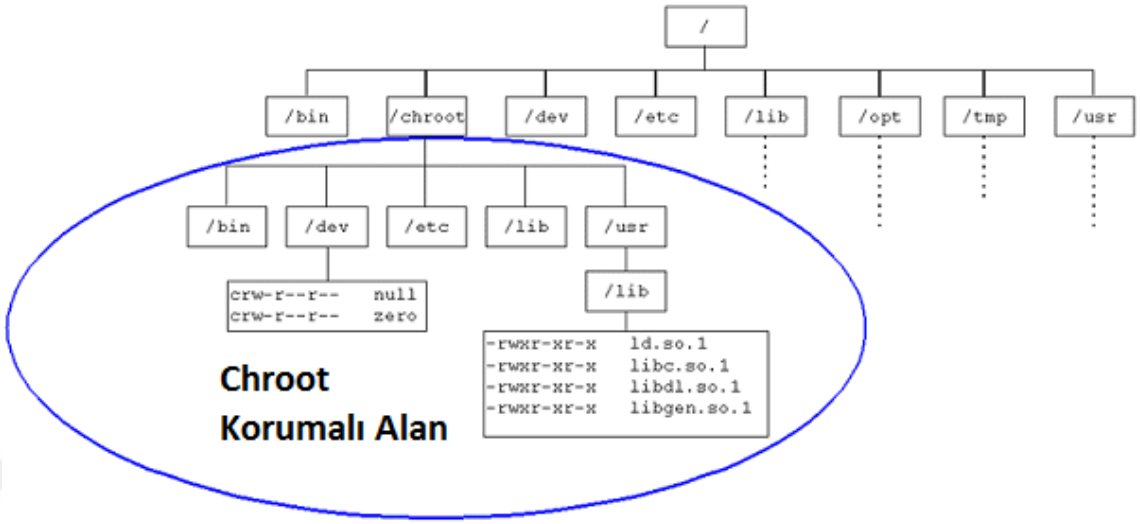
Şekil 5.6. Uygulamanın Sanal Alanda Olmadan(a) ve Sanal Alanda(b) Çalışması.

5.6. Chroot

Gentoo Wiki web sayfasında belirtildiği üzere("Chroot", 2019),

> (…)Chroot(Change root - Kökü değiştir), var olan kök dizinden farklı bir dizini kök dizin kabul ederek, ana sistemden mantıken bağımsız bir izole ortama geçiş yapmayı sağlayan bir Unix sistem aracıdır. Bu ortam "chroot hapishanesi" diye de anılır. Bu hapishanede çalışan kullanıcı, kendisine kök olarak tanımlanmış dizinin dışarısını göremez.

Chroot aracının ana kullanım alanlarından birisi Linux sistemi içerisinde test veya uyumluluk amaçlı farklı bir sistem ortamı yaratmaktır. Sanallaştırma yazılımı gerektirmediği için genelde sanallaştırmaya hafif bir alternatif olarak görülür.



Şekil 5.7. Chroot Uygulaması İle Yeni Dosya Sistemi Yapısı (Aridoshika, 2019).

Şekil 5.7'de görüldüğü gibi sunucu servis uygulamaların işletim sistemi kaynaklarına erişimi sınırlandırılmış olup sadece belirtilen ve gerekli kaynakların erişimine izin verilmektedir. Böylece sanal alan hale getirilmiş disk alanında çalışan uygulamalar diskin kalan kısımlarına hiç bir şekilde erişemezler. Bu yüzden chroot, web sunucuları ve alan ad sunucularına gelebilecek zararlı saldırıların dosya sisteminin geri kalanını etkilemesini önleyecek şekilde çalıştırılmaları ya da sistemin kalan kısmına zarar vermesinden korkulan uygulamaların denenmesi için oluşturulan sanal alanları canlandırmak için kullanılır (Kim & Zeldovich, 2010). Kurulan uygulamalar chroot dizini altında başka bir sistem üzerinde çalıştığını zanneder. chroot içerisinde çalıştırılacak uygulamalar statik olarak derlenir ve paylaşılan kütüphane bağımlılığı ortadan kaldırılır.

5.7. SELinux

Güvenlik Geliştirilmiş Linux (SELinux) C dilinde, 1999 yılında Amerika Birleşik Devletleri(ABD) Savunma Bakanlığı'na bağlı Ulusal Güvenlik Ajansı (NSA) tarafından ABD ordusu için geliştirilmiş ve daha sonrasında Linux çekirdeğine ilave edilmiş bir güvenlik kuralı parçasıdır (NSA, 2019). SELinux başta RedHat ve RedHat türevi olan Linux dağıtımlarında yerleşik olarak bulunmaktadır. Ancak her dağıtım bu

parçayı yerleşik olarak barındırmaz. Örneğin; Ubuntu dağıtımında bulunmaz ancak sonradan üzerine kurulum gerçekleştirilebilir.

SELinux sistem servislerine, kullanıcı programlarına, dosyalarına ve ağ kaynaklarına erişimi sınırlayan mecburi erişim kontrolü protokolüdür. İsteğe Bağlı Erişim Kontrolü (DAC) ve Zorunlu Erişim Kontrolü (MAC) olmak üzere iki modelde çalışmaktadır Çalışma prensibi temel olarak sistem ve kullanıcı servislerinde yapılması gereken işlemleri en az yetki ile kısıtlamayı amaçlamaktadır. DAC, sürecin ya da kullanıcının kaynaklara erişimini kullanıcı sahipliğine veya izinlerine bakarak belirlemektedir. MAC, kullanıcıların oluşturdukları nesnelerin üzerindeki denetim seviyesini, bütün dosya sistemi nesneleri için ilave etiket ekleyerek sınırlar ve MAC, DAC'a müdahalede bulunmaz. Dolayısıyla süreçlerin ve kullanıcıların söz konusu dosyalara erişebilmesi için etiketlere uygun erişim yetkilerinin olması gerekmektedir (Sueyasu, Tabata & Sakurai, 2003).

BÖLÜM 6

UYGULAMA PAKET FORMATLARININ KARŞILAŞTIRILMASI

Geleneksel ve modern paket formatları 4. Bölüm ’de açıklanmıştır. Geleneksel paket formatlarının bağımlılık yapısıyla çalıştığı bilinmektedir. Buna karşın modern paket formatlarının tüm bağımlılıklarını bir pakete sıkıştırdığından ve bu yönüyle nispeten sistemin geri kalanından izole bir şekilde çalışmalarını sürdürdüğünden bahsedilmiştir. Bu yeni nesil formatların ya da bu formatları yöneten paket yöneticilerinin dağıtımdan bağımsız bir şekilde çalışabileceğini ve hatta aynı işletim sistemi içinde uygulamaların farklı sürümlerini çalıştırabileceği aktarılmıştır.

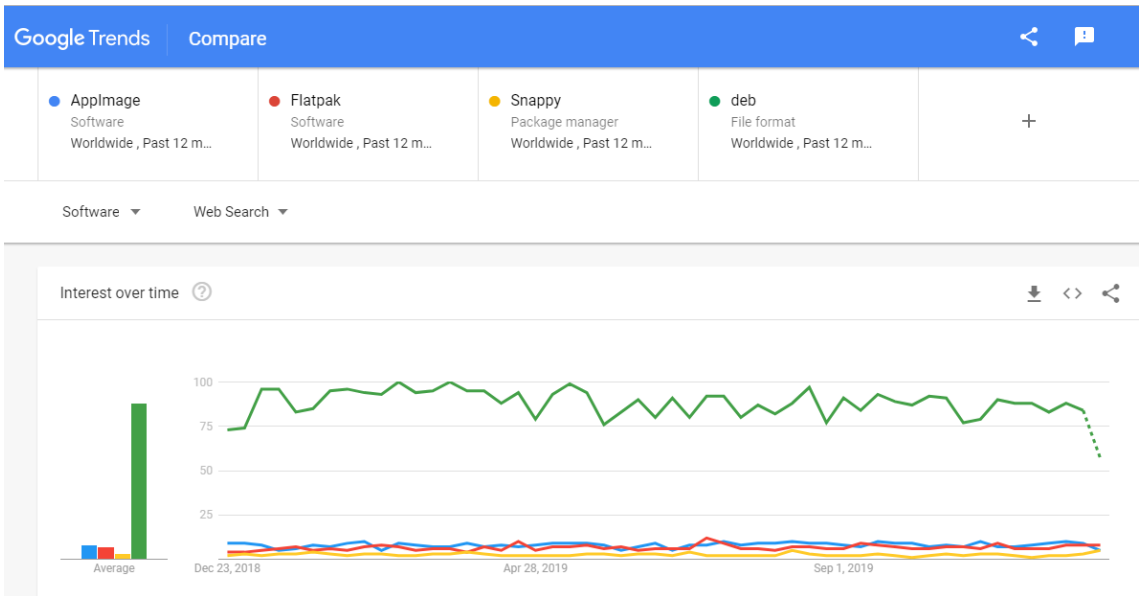
Uygulamaların sanal alanda çalışması ile ilgili bilgiler genel olarak Bölüm 5’te verilmiştir. Bu bölümde de bahsedilen geleneksel ve modern paket formatlarının sanal alanlarda çalıştırılabileceği gösterilmektedir. Bazı paket yöneticilerinde yerleşik olarak bulunan sanal alanda çalışma prensibi istenmesi halinde başka paket formatlarına da uygulanabilir. Modern paket formatlarının çalıştırıldıklarında bir sanal makine gibi davranması ve tüm bağımlılıklarını tek bir pakette bulundurmaları onları kendine özgü yapan yapıdır.

AppImage, Flatpak ve Snappy neredeyse tüm bağımlılıkları tek bir pakete sıkıştırdığından, açılan uygulama dosyaları geleneksel emsallerine göre yaklaşık dört kata kadar diskte daha fazla yer kaplarlar. Her ne kadar gelişen teknolojiler sayesinde depolama alanları sorun olmaktan çıksa da, yine de bu durum endişe verici olabilir. Doğuşundan bugüne asgari boyutta diskte yer kaplamayı amaçlayan geleneksel formatlara nazaran örneğin; indirilen 10 AppImage uygulama dosyasının, 10 kez aynı ya da benzer bağımlılıkları ilave ederek boyutlarının aşırı olması sorun olabilir (Verummeum, 2018).

Geleneksel formatların genellikle dağıtımlarla korunan resmi yazılım depolarına sahip olması bugüne kadar en büyük gücü olmuştur. Kullanıcılar için varsayılan bir paket yöneticisi ve yazılım merkezi bulunmaktadır ve her zaman bir paket arayabilecekleri bir platform olması kullanıcılara güven vermektedir. Kullanıcılar kullanacakları dağıtımları seçerken paket yöneticilerine de dikkat ederler. Bunun aksine şimdilik yeni nesil paket formatları bu bakımdan zayıf kalmaktadır. Herhangi bir uygulamanın AppImage, Flatpak ya da Snappy formatında arayıp bulması zor olabilir.

Uygulama üreticileri kullanıcılarına web sitelerinde birçok uygulamanın farklı formatlarda en yeni sürümlerini sunmaya devam ediyorlar. Bir takım kontrol işlemlerinden sonra da yazılım depolarına eklenebiliyor. Bu durum yeni uygulama sürümlerini çabuk almak isteyen kimi kullanıcılar için harika olurken Uzun Süreli Destek (LTS) dağıtımı kullanan kimi kullanıcılar için öyle olmayabilir. Bu kullanıcılar güncelleme ile yeni sürümleri alamazlar. Diğer taraftan geleneksel formatlar için yeni sürüm çalışmaları yapan üreticiler AppImage, Flatpak ve Snappy gibi yeni formatlara geçmeleri zaman alacaktır.

Dünya genelinde Şekil 6.1'de Google arama motorundaki isim olarak web aramalarına bakıldığında geleneksel paket formatı "deb" in modern paket formatlarına göre ezici üstünlüğü görünmektedir.



Şekil 6.1. AppImage, Flatpak, Snappy ve deb Arama Trendleri[4] (Google Trends,2019).

Modern paket formatları programlama ve otomasyon araçları ile neredeyse hiç etkileşime girmezler. Bunlardan bir AppImage uygulamasını dağıtım sisteminde farklı görevler için kullanmak istendiğinde ya da yapılandırmak istendiğinde zorluk çekilecektir. Bununla birlikte Flatpak ve Snappy uygulamaları sistemin geri kalanından izole bir şekilde çalıştığından yine bunlarda da yapılandırma işlemleri zor olacaktır. Bu durumların modern paket formatları için dezavantajlı olmasıyla birlikte zaten geleneksel paketlerde bu durum iyi durumda değildir.

AppImage ve Flatpak masaüstü kullanıcıları düşünülerek oluşturulurken Snappy gömülü aygıtlar dâhil olmak üzere masaüstü bilgisayarlar, sunucular, IoT cihazlar gibi çeşitli aygıtlarda çalışabilir. Bunların yanında Debian formatı ise masaüstü bilgisayarlar ve sunucular için vazgeçilmez konumdadır.

Modern paket formatları AppImage, Flatpak ve Snappy ile geleneksel paket formatları olan Debian türevi dağıtımların deb ve Red Hat türevi dağıtımların varsayılan paket formatı rpm formatlarına ait önemli özellik bilgileri Çizelge 6.1'de verilmiştir ("Similar projects", 2019).

[4] Interest over time(Zaman içindeki ilgi) Sayılar, verilen bölge ve süre için grafikteki en yüksek noktaya göre arama ilgisini gösterir. 100 değeri, terimin en yüksek popülerlik olduğunu gösterir. 50 değeri, terimin yarı popüler olduğu anlamına gelir. 0 puan, bu terim için yeterli veri olmadığı anlamına gelir.

Çizelge 6.1. Modern ve Geleneksel Paket Formatlarının Önemli Özellik Bilgileri (EDITOR, 2019).

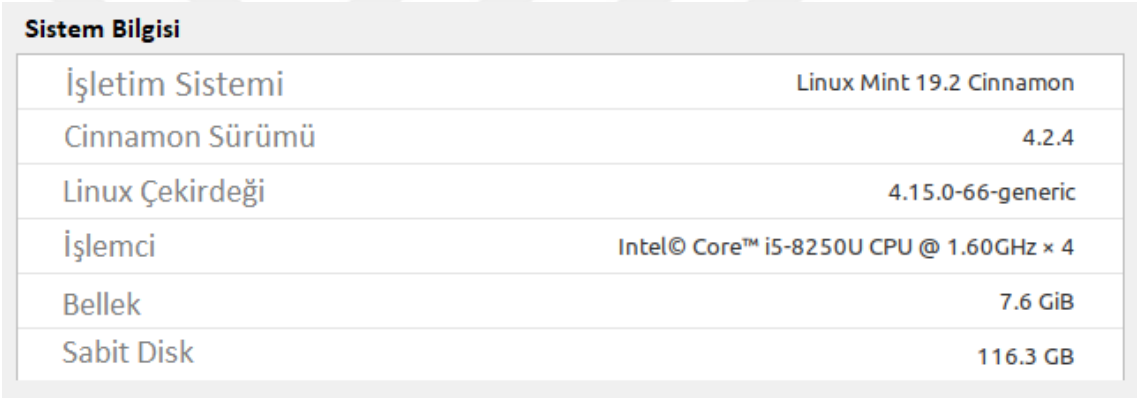
Özellik	**AppImage**	**Flatpak**	**Snappy**	**deb**	**rpm**
Benzersiz Özellik	Bir uygulama mağazası veya depoya sahip değildir, sadece tüm dağıtımlar için paketlenmiş bir dosya mevcuttur.	FlatHub adında bir uygulama mağazasına sahiptir, ancak kullanıcılar paketleri barındırabilir ve dağıtabilirler.	Canonical Ltd. tarafından yönetilen (Ubuntu'nun da bağlı olduğu), merkezi uygulama deposuna ve Canonical'ın aktif katkısına sahiptir.	Debian ve türevi Linux dağıtımlarında kullanılan ana paket formatıdır.	Red Hat ve türevi Linux dağıtımlarında kullanılan ana paket formatıdır.
Hedef Sistemi	Masaüstü bilgisayarlar ve sunucular.	Masaüstü bilgisayarlar ve sunucular.	Masaüstü bilgisayarlar, sunucular, IoT cihazlar, gömülü cihazlar vb.	Masaüstü bilgisayarlar ve sunucular.	Masaüstü bilgisayarlar ve sunucular.
Kütüphaneler/ Bağımlılıklar	Kütüphaneler ve diğer bağımlılıklar paketlenmiştir.	GNOME, KDE, Freedesktop veya özel olarak paketlenmiştir.	Temel sistem veya eklentiler aracılığıyla veya paketlenebilir.	Kütüphaneler ve diğer bağımlılıklar ayrıdır.	Kütüphaneler ve diğer bağımlılıklar ayrıdır.
Geliştiriciler	Simon Peter tarafından yönetilen topluluk.	Flatpak ekibi tarafından desteklenen topluluk.	Canonical Ltd. Şirketi.	Girişim tarafından desteklenen topluluk.	Girişim ve Red Hat tarafından desteklenen topluluk.
Programlama Dili	C.	C.	Golang, C ve Python.	C.	C, Perl.
İlk Sürüm Tarihi	2004.	2015.	2014.	1995.	1997.
Kullandığı Dosya Sistemi	SQUASHFS	OSTree	SQUASHFS	Ext4	Ext4
Sandboxing(Sanal Alan)	Uygulanabilir.	İzole ancak uygulamaları varsayılan olarak çalıştırmak için sistem dosyalarını kullanır.	Değişken sınırlandırma özelliklerine sahip katı, klasik ve donanım türü var. İzolasyonla çalışır.	Uygulanabilir.	Uygulanabilir.
Sandboxing(Sanal Alan) Platformu	Firejail, AppArmor, Bubblewrap.	Bubblewrap.	AppArmor.	Firejail, AppArmor, Bubblewrap.	Firejail, AppArmor, Bubblewrap.
Merkezi Depo	AppImageHub.	Flathub.	Snapcraft Mağazası.	Resmi ve özel depolar.	Resmi ve özel depolar.

Çizelge 6.1. Modern ve Geleneksel Paket Formatlarının Önemli Özellik Bilgileri(Devamı).

Uygulama Kurulumu	Gerekli değil. Taşınabilir uygulama olarak kullanılır.	Flatpak paket yöneticisi kullanarak kurulum yapılır.	Snapd paket yöneticisi kullanarak kurulum yapılır.	Sistemdeki varsayılan paket yöneticisi kullanılarak kurulum yapılır.	Sistemdeki varsayılan paket yöneticisi kullanılarak kurulum yapılır.
Uygulama Yürütme	Çalıştırılabilir dosya olarak ayarladıktan sonra kullanılabilir.	CLI kullanılıyorsa flatpak komutuyla çalıştırılması gerekiyor.	Masaüstü tümleşik ek araçlarını kullanabilir.	Masaüstü tümleşik ek araçlarını kullanabilir.	Masaüstü tümleşik ek araçlarını kullanabilir.
Paket Kurulumu İçin Kök Erişim Yetkisi	Kök kullanıcı erişim yetkisi olmadan çalıştırılabilir.	Kurulum için kök kullanıcı erişim yetkisi gerekli.	Kurulum için kök kullanıcı erişim yetkisi bazen gerekli.	Kurulum için kök kullanıcı erişim yetkisi gerekli.	Kurulum için kök kullanıcı erişim yetkisi gerekli.
Barınma	Herhangi biri tarafından herhangi bir yerde barınabilir.	Herhangi biri tarafından herhangi bir yerde barınabilir.	Tescilli olan Canonical sunucularında barındırılır.	Herhangi biri tarafından herhangi bir yerde barınabilir.	Herhangi biri tarafından herhangi bir yerde barınabilir.
Sistem Dışı Konumlardan Taşınabilir Yürütme	Evet.	Evet, flatpak istemcisi yapılandırıldıktan sonra.	Hayır.	Evet.	Evet.
Uygulamanın birden çok sürümünü çalıştırma	Mümkün.	Mümkün.	Tek oturumda uygulamanın bir sürümü. Daha fazlası için ayrı ayrı yapılandırılmalıdır.	Mümkün değil.	Mümkün değil.
Uygulamaları Güncelleme	CLI kullanarak AppImageUpdate veya AppImage içinde yerleşik bir güncelleme aracı aracılığıyla.	Güncelleme Flatpak güncelleme komutunu kullanarak.	Güncelleme destekler, otomatik olarak güncellenir.	Sistem içinde yerleşik bir güncelleme aracı aracılığıyla.	Sistem içinde yerleşik bir güncelleme aracı aracılığıyla.
Masaüstü bilgisayar dışında çalışma	Evet	Hayır	Evet	Evet	Evet
Dağıtım oranı	Düşük	Orta	Orta	Yüksek	Yüksek

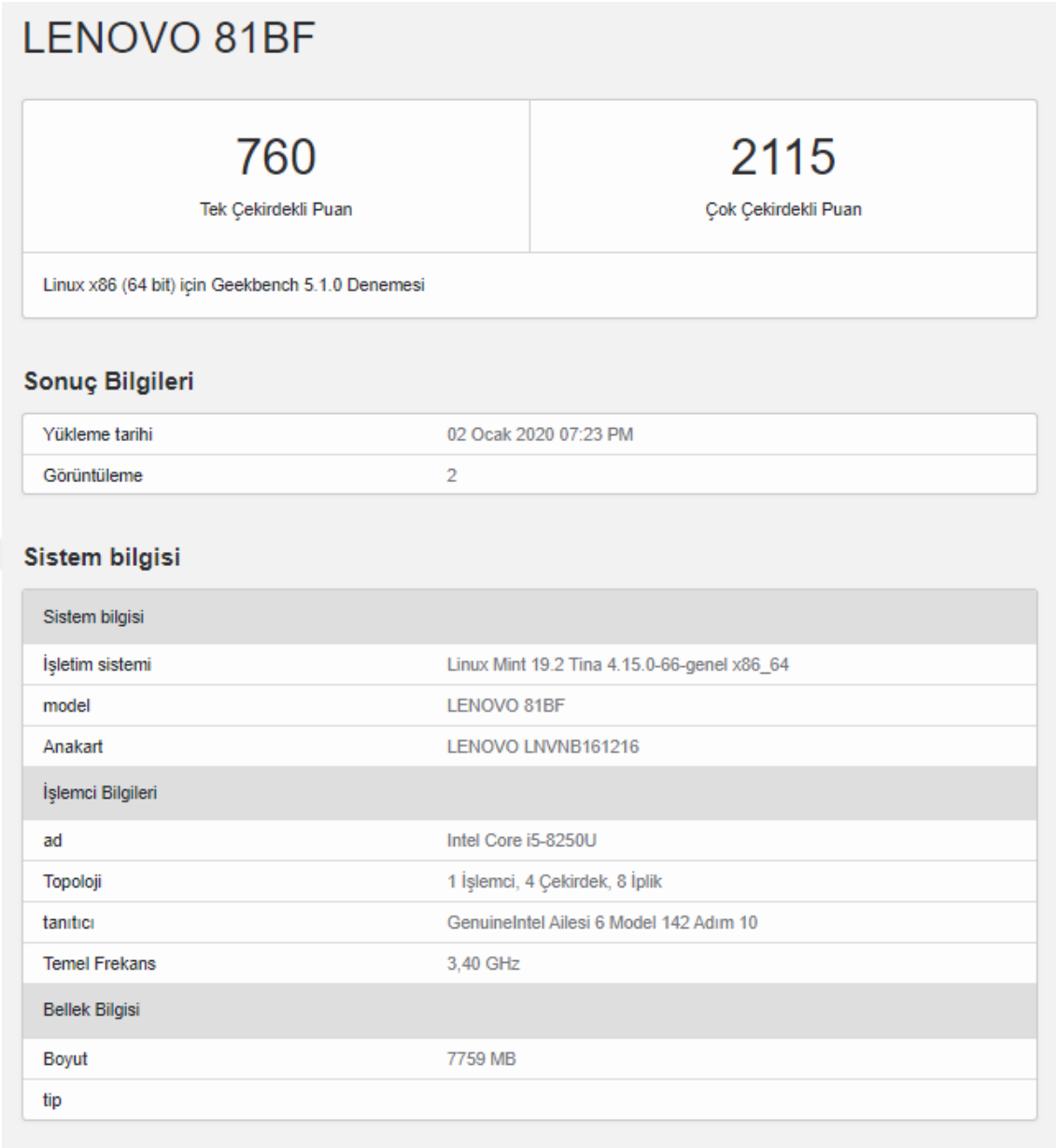
6.1. Linux Mint Dağıtımında Örnek Uygulamaların Paket Formatları Açısından Karşılaştırılması

Bu alt bölümde, paket formatları arasında önemli farklılıklar olup olmadığını görmek için Linux Mint 19.2 dağıtımında popüler medya oynatıcısı olan Vlc, çeşitli ofis uygulamalarını içeren Libreoffice ve video düzenleme uygulaması olan Kdenlive paketleri AppImage, Flatpak, Snappy ve deb formatlarında boyut ve performans olarak incelendi. Paket formatlarına ait boyut bilgileri paket yöneticilerinin vermiş olduğu bilgiler ve inceleme bilgisayarının disk kullanım alanları birlikte değerlendirilerek elde edilmiştir. Paket formatlarının karşılaştırıldığı bilgisayar sisteminin özellikleri Şekil 6.2'de, bu bilgisayar sistemine ait Geekbench[5] performans sonuçları ise Şekil 6.3'de verilmiştir.



Şekil 6.2. Linux Mint Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Özellikleri(Linux Mint, 2019).

[5] Geekbench, tek çekirdekli ve çok çekirdekli performansı birbirinden ayıran bir puanlama sistemi ve gerçek dünya senaryolarını simüle eden iş yüklerine sahip bir çapraz platform işlemci ölçütüdür (Norf, 2019).



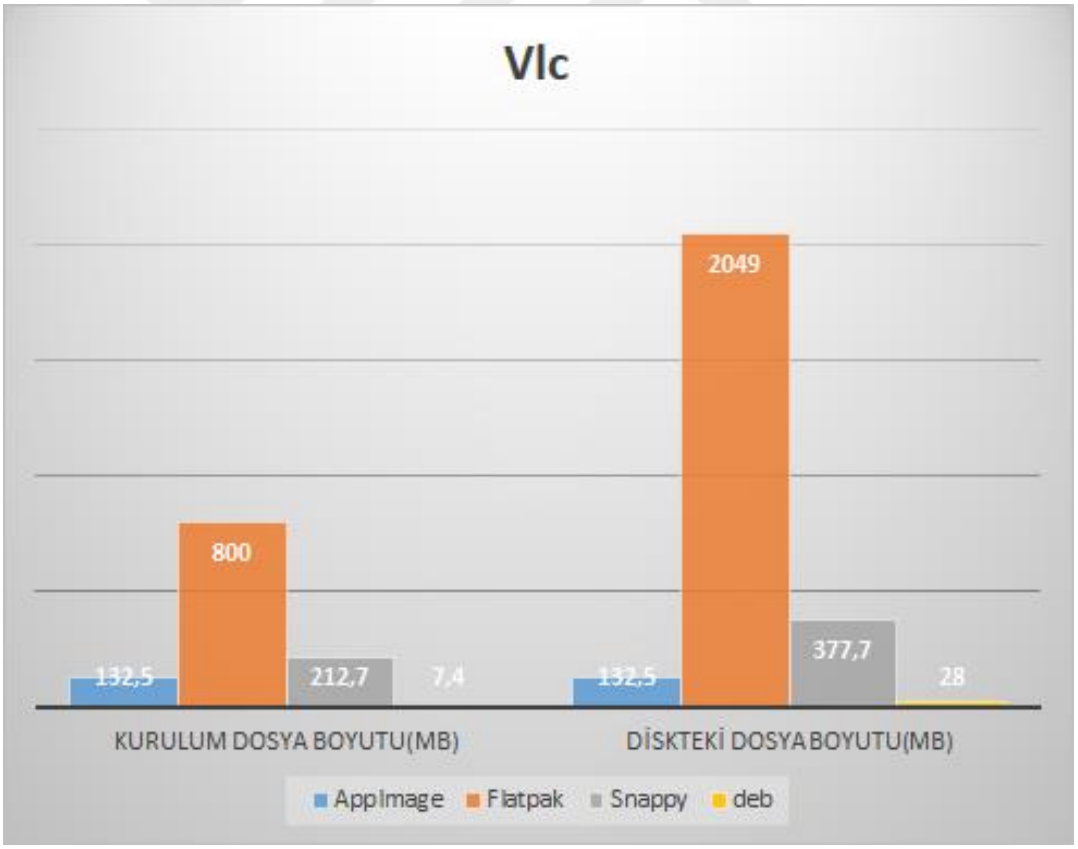
Şekil 6.3. Linux Mint Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Geekbench Performans Sonuçları(Geekbench Browser, 2019).

6.1.1. Vlc

Vlc uygulamasının 3.0.8 sürümünde her bir paket formatı için arka arkaya 5 tekrarlı olarak yaklaşık 3,5 GB boyutunda mp4 uzantılı video dosyası genel olarak çalışma performansı gözlemlendi. AppImage dışındaki diğer paket formatlarında bir farklılık gözlemlenmedi ancak AppImage ses dosyalarını açarken video dosyalarını açamayıp kapandı. Bunlara ilişkin karşılaştırma bilgileri Çizelge 6.2'de ve Şekil 6.4'de verilmiştir.

Çizelge 6.2. Linux Mint Dağıtımında Vlc Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları(İlk açılma süresi saniye olarak verilmiştir).

Özellik	**AppImage**	**Flatpak**	**Snappy**	**deb**
Çalışabilirlik	Evet	Evet	Evet	Evet
Kurulum Dosya Boyutu(MB)	132,5	800	212,7	7,4
Diskteki Dosya Boyutu(MB)	132,5	2049	377,7	28
İlk Açılma Süresi(sn)	4,5	2,5	2	1,5
Çalışma Performansı	Ses dosyası açıyor ancak video açmıyor.	Hızlı	Hızlı	Hızlı
Kaldırılabilme	Evet	Evet	Evet	Evet



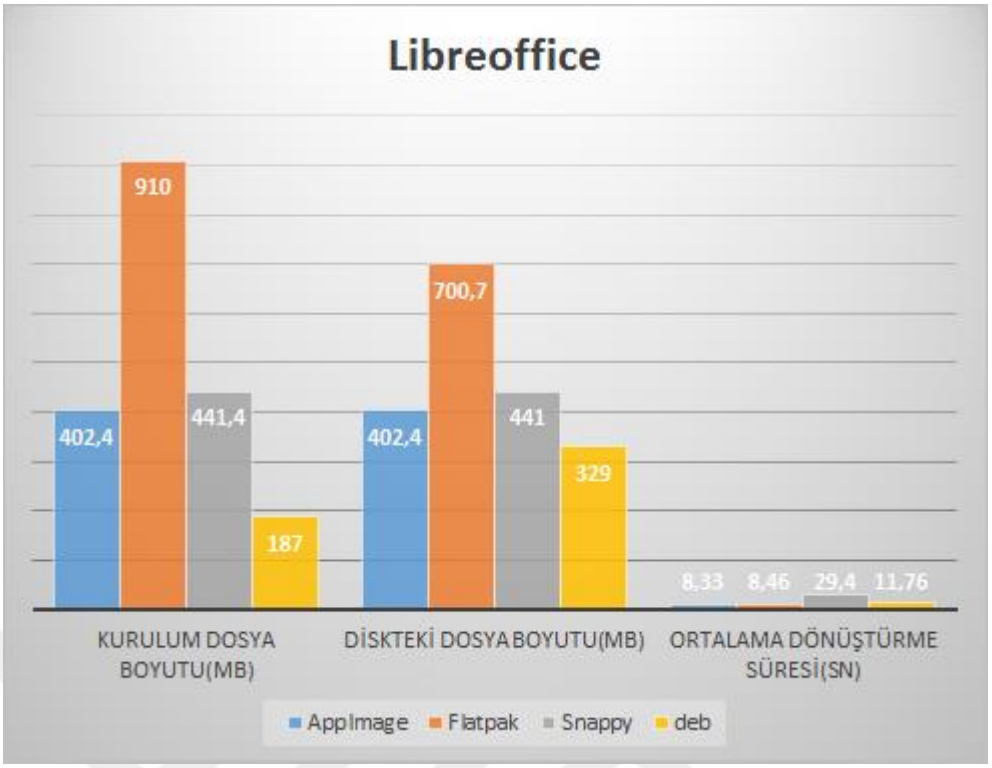
Şekil 6.4. Linux Mint Dağıtımında Vlc Uygulamasının Farklı Paket Formatlarında Disk Boyutu.

6.1.2. Libreoffice

Libreoffice uygulamasının 6.3.3 sürümünde her bir paket formatı için arka arkaya 5 tekrarlı olarak 220 sayfa pptx uzantılı sunum dosyası % 100 kalite ve 600 İnç Başına Nokta (DPI) olacak şekilde Taşınabilir Belge Biçimi'ne (PDF) dönüştürmek için geçen süre gözlemlenmiştir. Ortalama sonuçlar Çizelge 6.3'de verilmiştir. Ayrıca bu formatlara ait diskte kapladıkları boyut ve işlem performansı Şekil 6.5'de görünmektedir.

Çizelge 6.3. Linux Mint Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları(İlk açılma ve dönüştürme süresi saniye olarak verilmiştir).

Özellik	**AppImage**	**Flatpak**	**Snappy**	**deb**
Çalışabilirlik	Evet	Evet	Evet	Evet
Kurulum Dosya Boyutu (MB)	402,4	910	441,4	187
Diskteki Dosya Boyutu (MB)	402,4	700,7	441	329
İlk Açılma Süresi(sn)	2,3	2,1	2,7	2
Çalışma Performansı	Hızlı	Hızlı	Orta	Hızlı
Ortalama Dönüştürme Süresi (sn)	8,33	8,46	29,4	11,76
Kaldırılabilme	Evet	Evet	Evet	Evet



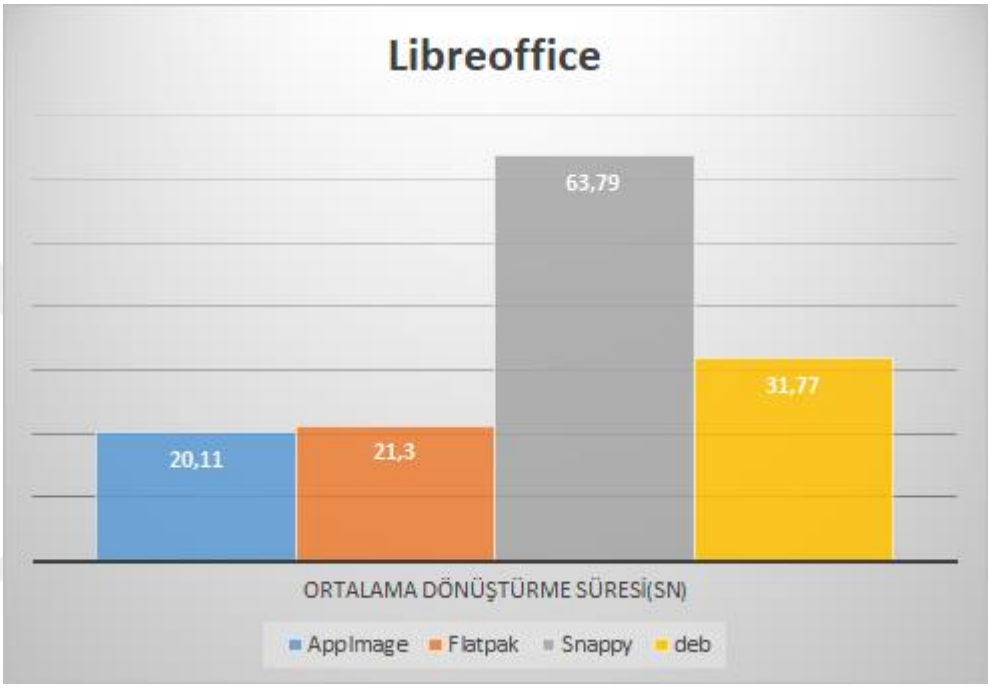
Şekil 6.5. Linux Mint Dağıtımında Libreoffice Uygulamasının Farklı Paket Formatlarında Disk Boyutu ve İşlem Performansı.

6.1.2.1. Libreoffice Uygulamasının Ağır Yük Altındaki Performansı

Linux Mint dağıtımında incelenen uygulama formatlarında Libreoffice uygulamasının ağır yük altındaki performansını ölçmek için bir önceki alt bölümdeki sunum dosyası dönüştürme testi her formatın Kdenlive video düzenleme uygulaması ile birlikte tekrar gerçekleştirilmiştir. Bilgisayar sisteminin işlemci ve ana bellek kullanımına yük bindirmek amacıyla bir sonraki alt bölümde verilen Kdenlive uygulama testi sistemde sürekli çalışır durumda bırakılmıştır. Bu yöntemle dosya dönüştürme testinde kullanılan işlemci kullanım yüküne göre yaklaşık 10 kat bir artış sağlanmıştır. Bu sonuç, dağıtımın CLI'de gerçek zamanlı olarak uygulamaların sistemdeki kaynak kullanımını ve çalışan iş parçacıklarının listesini gösteren top komutu aracılığı ile yapılan gözlemler sayesinde elde edilmiştir. Ortalama sonuçlar Çizelge 6.4'de ve Şekil 6.6'da verilmiştir.

Çizelge 6.4. Linux Mint Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında Ağır Yük Altında İnceleme Sonuçları(Dönüştürme süresi saniye olarak verilmiştir).

	AppImage	**Flatpak**	**Snappy**	**deb**
Ortalama Dönüştürme Süresi (sn)	20,11	21,30	63,79	31,77



Şekil 6.6. Linux Mint Dağıtımında Libreoffice Uygulamasının Farklı Paket Formatlarında Ağır Yük Altında İşlem Performansı.

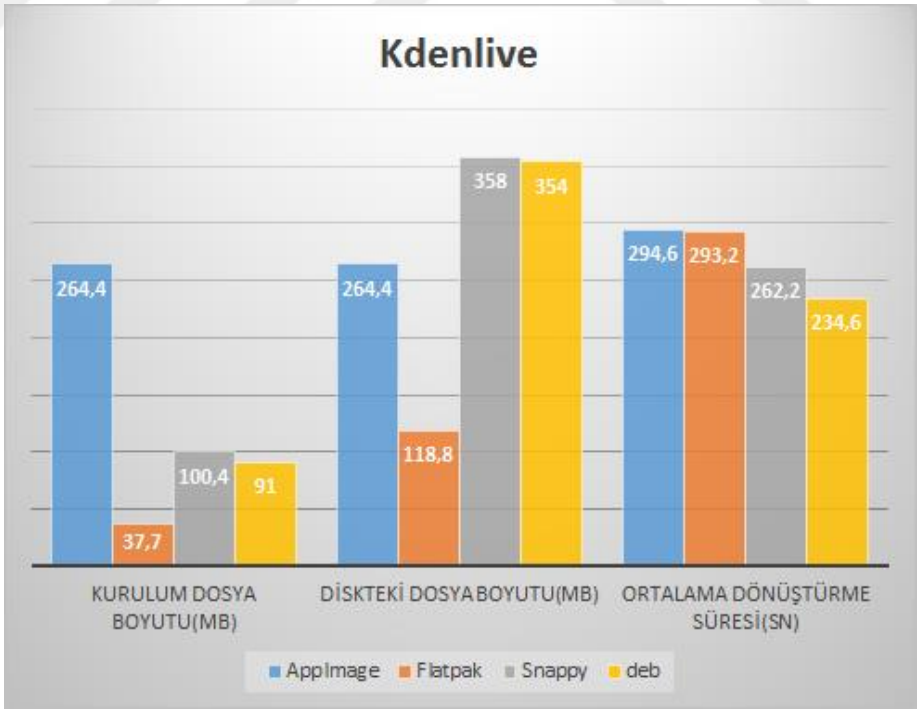
Ağır yük altında her formatta Libreoffice uygulamasının dosya dönüştürme süreleri Şekil 6.6'da da görüldüğü gibi bir önceki teste göre yaklaşık 2 kat artmıştır.

6.1.3. Kdenlive

Kdenlive uygulamasının 19.08 sürümünde her bir paket formatı için arka arkaya 5 tekrarlı olarak 500 MB boyutunda ham video kamera görüntü dosyası bir Apple Inc tarafından geliştirilen ve MPEG-4 sıkıştırma kodunu kullanan MOV formatından yaygın olarak kullanılan ve H.265/HEVC sıkıştırma kodunu kullanan MP4 formatına dönüştürmek için geçen süre gözlemlenmiştir. Ortama sonuçlar aşağıda Çizelge 6.5'de dönüştürme süresi saniye olarak verilmiştir. Ayrıca bu formatlara ait diskte kapladıkları boyut ve işlem performansı Şekil 6.7'de gösterilmektedir.

Çizelge 6.5. Linux Mint Dağıtımında Kdenlive Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları(İlk açılma ve dönüştürme süresi saniye olarak verilmiştir).

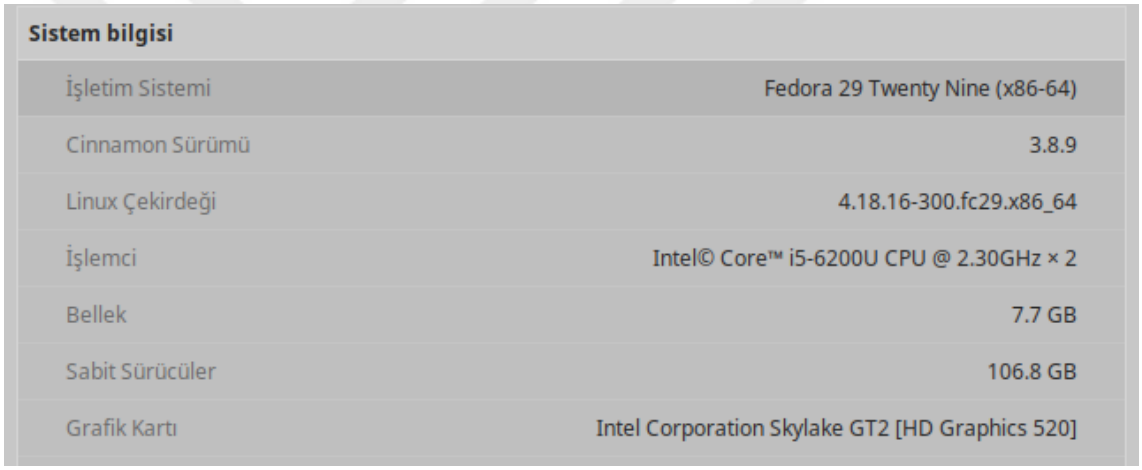
Özellik	**AppImage**	**Flatpak**	**Snappy**	**deb**
Çalışabilirlik	Evet	Evet	Evet	Evet
Kurulum Dosya Boyutu (MB)	264,4	37,7	100,4	91
Diskteki Dosya Boyutu (MB)	264,4	118,8	358	354
İlk Açılma Süresi(sn)	3	2,5	2	2
Çalışma Performansı	Orta	Hızlı	Hızlı	Hızlı
Ortalama Dönüştürme Süresi (sn)	294,6	293,2	262,2	234,6
Kaldırılabilme	Evet	Evet	Evet	Evet



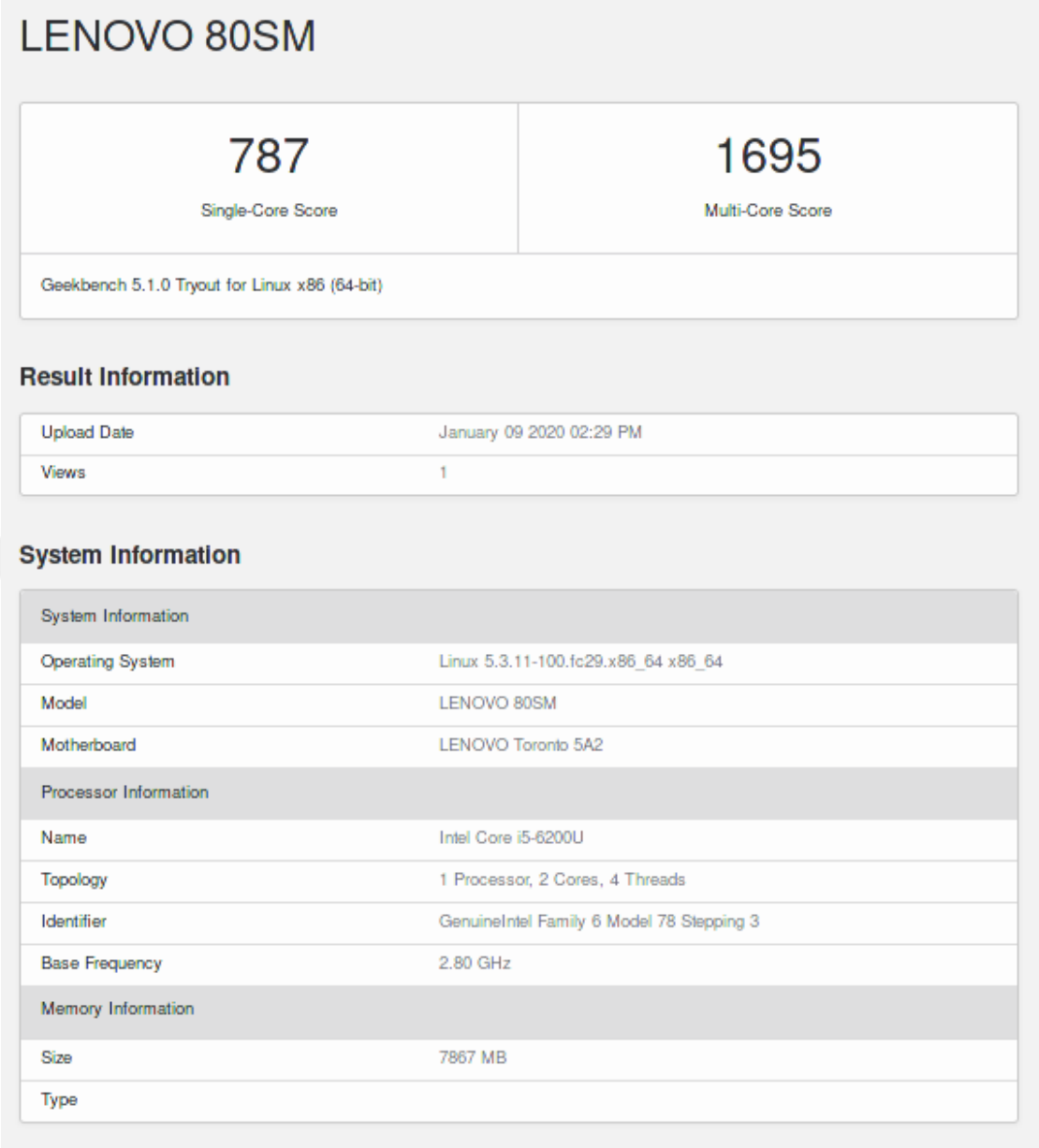
Şekil 6.7. Linux Mint Dağıtımında Kdenlive Uygulamasının Farklı Paket Formatlarında Disk Boyutu ve İşlem Performansı.

6.2. Fedora Dağıtımında Örnek Uygulamaların Paket Formatları Açısından Karşılaştırılması

Bu alt bölümde, paket formatları arasında önemli farklılıklar olup olmadığını görmek için Fedora 29.1.2 dağıtımında popüler medya oynatıcısı olan Vlc, çeşitli ofis uygulamalarını içeren Libreoffice ve video düzenleme uygulaması olan Kdenlive paketleri AppImage, Flatpak, Snappy ve rpm formatlarında boyut ve performans olarak incelendi. Paket formatlarına ait boyut bilgileri paket yöneticilerinin vermiş olduğu bilgiler ve inceleme bilgisayarının disk kullanım alanları birlikte değerlendirilerek elde edilmiştir. Paket formatlarının karşılaştırıldığı bilgisayar sisteminin özellikleri Şekil 6.8'de, bu bilgisayar sistemine ait Geekbench performans sonuçları ise Şekil 6.9'da verilmiştir.



Şekil 6.8. Fedora Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Özellikleri(Fedora, 2019).



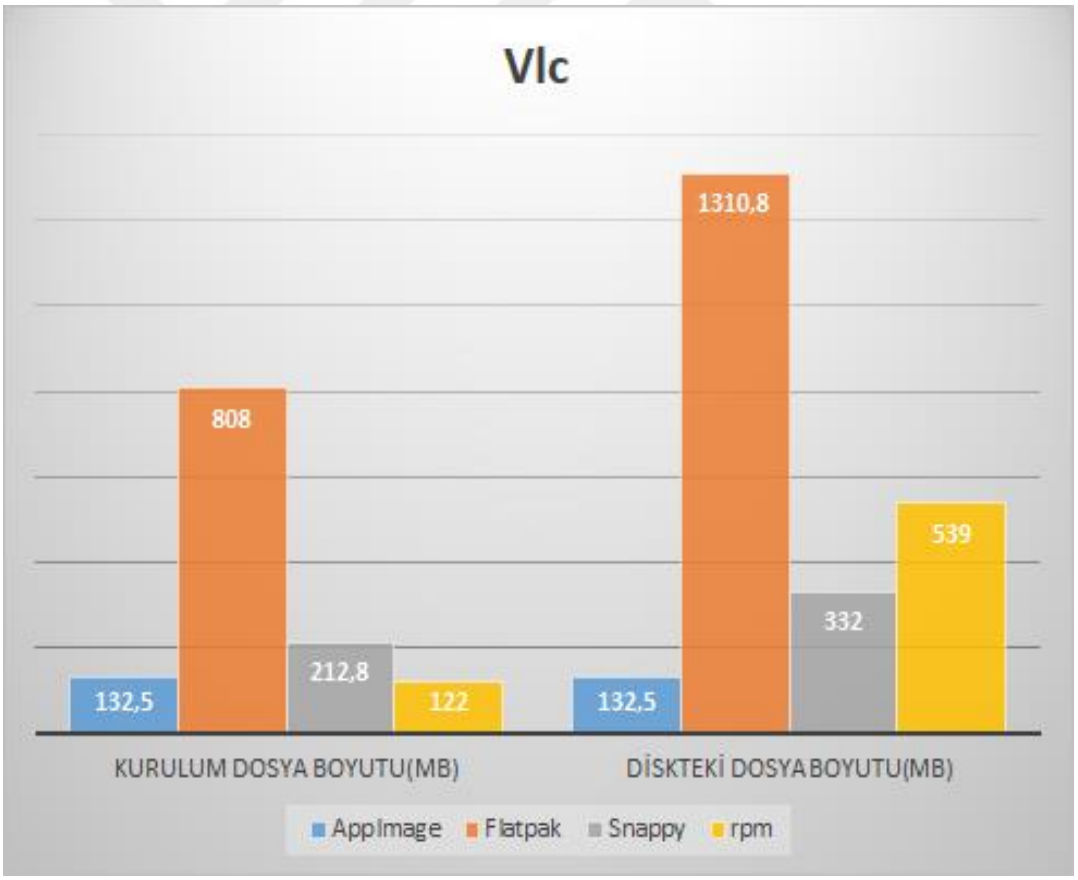
Şekil 6.9. Fedora Dağıtımında Paket Formatlarının Karşılaştırıldığı Bilgisayar Sisteminin Geekbench Performans Sonuçları(Geekbench Browser, 2019).

6.2.1. Vlc

Vlc uygulamasının 3.0.8 sürümünde her bir paket formatı için arka arkaya 5 tekrarlı olarak yaklaşık 3,5 GB boyutunda mp4 uzantılı video dosyası genel olarak çalışma performansı gözlemlendi. AppImage dışındaki diğer paket formatlarında bir farklılık gözlemlenmedi ancak AppImage Vlc uygulaması çalışmadı. Bunlara ilişkin karşılaştırma bilgileri Çizelge 6.6'da ve Şekil 6.10'da verilmiştir.

Çizelge 6.6. Fedora Dağıtımında Vlc Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları(İlk açılma süresi saniye olarak verilmiştir).

Özellik	**AppImage**	**Flatpak**	**Snappy**	**rpm**
Çalışabilirlik	Hayır	Evet	Evet	Evet
Kurulum Dosya Boyutu(MB)	132,5	800	212,8	122
Diskteki Dosya Boyutu(MB)	132,5	1310,8	332	539
İlk Açılma Süresi(sn)	Yok	0,4	2,5	0,5
Çalışma Performansı	Yok	Hızlı	Orta	Hızlı
Kaldırılabilme	Evet	Evet	Evet	Evet



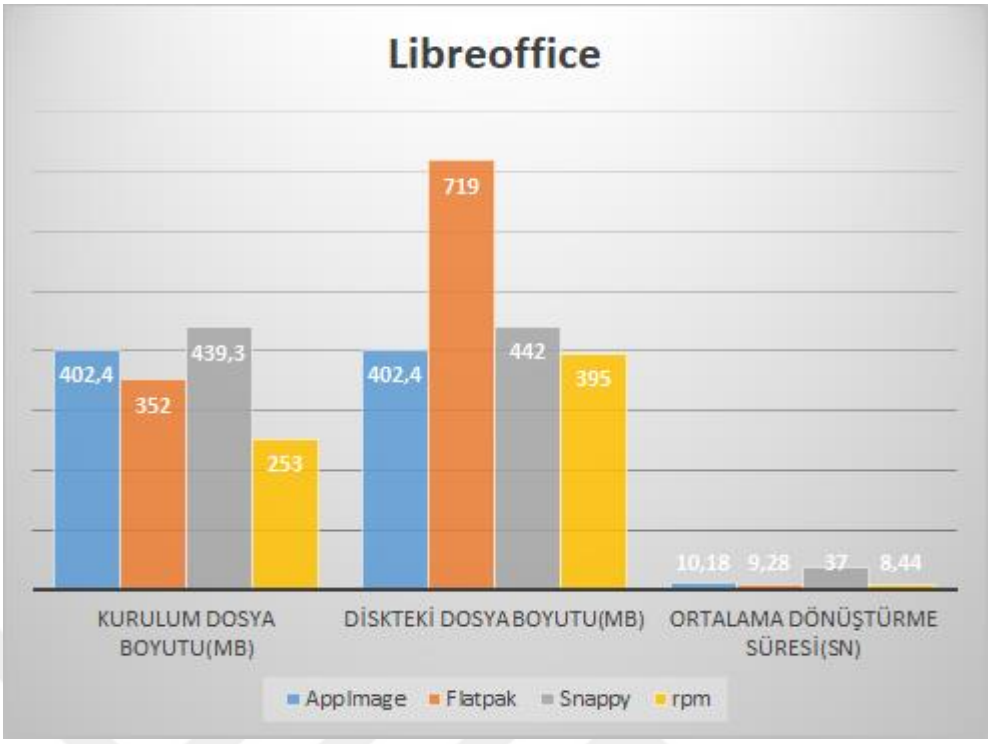
Şekil 6.10. Fedora Dağıtımında Vlc Uygulamasının Farklı Paket Formatlarında Disk Boyutu.

6.2.2. Libreoffice

Libreoffice uygulamasının 6.3.3 sürümünde her bir paket formatı için arka arkaya 5 tekrarlı olarak 220 sayfa pptx uzantılı sunum dosyası % 100 kalite ve 600 İnç Başına Nokta (DPI) olacak şekilde Taşınabilir Belge Biçimi'ne (PDF) dönüştürmek için geçen süre gözlemlenmiştir. Ortalama sonuçlar Çizelge 6.7'de verilmiştir. Ayrıca bu formatlara ait diskte kapladıkları boyut ve işlem performansı Şekil 6.11'de görünmektedir.

Çizelge 6.7. Fedora Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları(İlk açılma ve dönüştürme süresi saniye olarak verilmiştir).

Özellik	**AppImage**	**Flatpak**	**Snappy**	**rpm**
Çalışabilirlik	Evet	Evet	Evet	Evet
Kurulum Dosya Boyutu (MB)	402,4	352	439,3	253
Diskteki Dosya Boyutu (MB)	402,4	719	442	395
İlk Açılma Süresi(sn)	3,5	1	5	1,5
Çalışma Performansı	Hızlı	Hızlı	Orta	Hızlı
Ortalama Dönüştürme Süresi (sn)	10,18	9,28	37	8,44
Kaldırılabilme	Evet	Evet	Evet	Evet



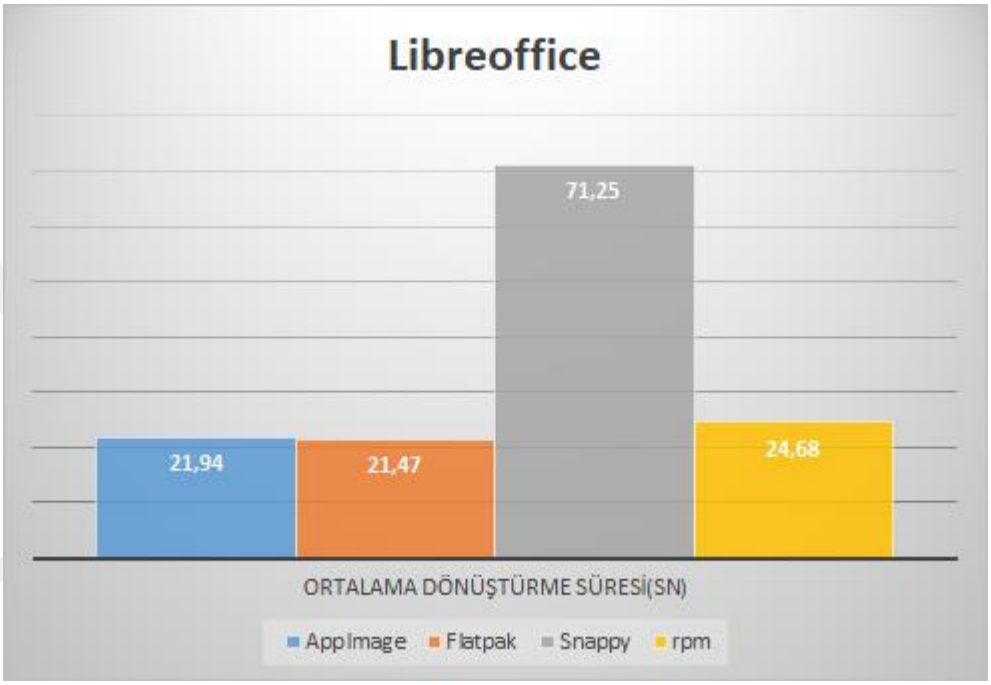
Şekil 6.11. Fedora Dağıtımında Libreoffice Uygulamasının Farklı Paket Formatlarında Disk Boyutu ve İşlem Performansı.

6.2.2.1. Libreoffice Uygulamasının Ağır Yük Altındaki Performansı

Fedora dağıtımında incelenen uygulama formatlarında Libreoffice uygulamasının ağır yük altındaki performansını ölçmek için bir önceki alt bölümdeki sunum dosyası dönüştürme testi her formatın Kdenlive video düzenleme uygulaması ile birlikte tekrar gerçekleştirilmiştir. Bilgisayar sisteminin işlemci ve ana bellek kullanımına yük bindirmek amacıyla bir sonraki alt bölümde verilen Kdenlive uygulama testi sistemde sürekli çalışır durumda bırakılmıştır. Bu yöntemle dosya dönüştürme testinde kullanılan işlemci kullanım yüküne göre yaklaşık 10 kat bir artış sağlanmıştır. Bu sonuç, dağıtımın CLI'de gerçek zamanlı olarak uygulamaların sistemdeki kaynak kullanımını ve çalışan iş parçacıklarının listesini gösteren top komutu aracılığı ile yapılan gözlemler sayesinde elde edilmiştir. Ortalama sonuçlar Çizelge 6.8'de ve Şekil 6.12'de verilmiştir.

Çizelge 6.8. Fedora Dağıtımında Libreoffice Uygulamasının Geleneksel ve Modern Paket Formatlarında Ağır Yük Altında İnceleme Sonuçları(Dönüştürme süresi saniye olarak verilmiştir).

	AppImage	**Flatpak**	**Snappy**	**rpm**
Ortalama Dönüştürme Süresi (sn)	21,94	21,47	71,25	24,68



Şekil 6.12. Fedora Dağıtımında Libreoffice Uygulamasının Farklı Paket Formatlarında Ağır Yük Altında İşlem Performansı.

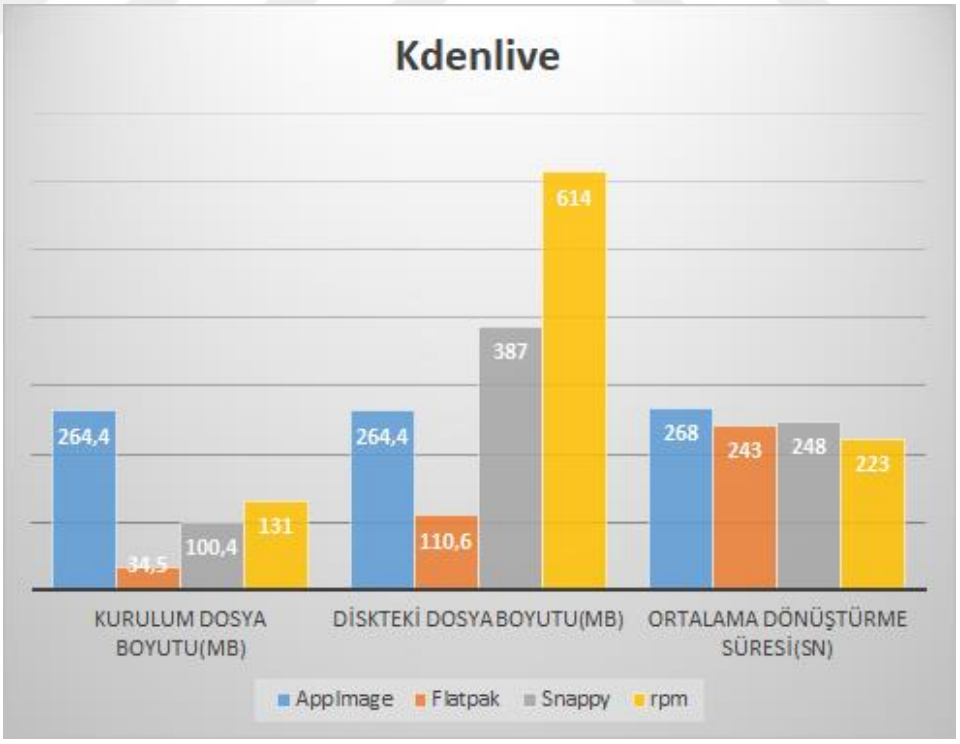
Ağır yük altında her formatta Libreoffice uygulamasının dosya dönüştürme süreleri Şekil 6.12'de de görüldüğü gibi bir önceki teste göre yaklaşık 2 kat artmıştır.

6.2.3. Kdenlive

Kdenlive uygulamasının 19.08 sürümünde her bir paket formatı için arka arkaya 5 tekrarlı olarak 500 MB boyutunda ham video kamera görüntü dosyası bir Apple Inc tarafından geliştirilen ve MPEG-4 sıkıştırma kodunu kullanan MOV formatından yaygın olarak kullanılan ve H.265/HEVC sıkıştırma kodunu kullanan MP4 formatına dönüştürmek için geçen süre gözlemlenmiştir. Ortama sonuçlar aşağıda Çizelge 6.9'da dönüştürme süresi saniye olarak verilmiştir. Ayrıca bu formatlara ait diskte kapladıkları boyut ve işlem performansı Şekil 6.13'de gösterilmektedir.

Çizelge 6.9. Fedora Dağıtımında Kdenlive Uygulamasının Geleneksel ve Modern Paket Formatlarında İnceleme Sonuçları(İlk açılma ve dönüştürme süresi saniye olarak verilmiştir).

Özellik	**AppImage**	**Flatpak**	**Snappy**	**rpm**
Çalışabilirlik	Evet	Evet	Evet	Evet
Kurulum Dosya Boyutu (MB)	264,4	34,5	100,4	131
Diskteki Dosya Boyutu (MB)	264,4	110,6	387	614
İlk Açılma Süresi(sn)	2,5	2	4	0,5
Çalışma Performansı	Orta	Hızlı	Hızlı	Hızlı
Ortalama Dönüştürme Süresi (sn)	268	243	248	223
Kaldırılabilme	Evet	Evet	Evet	Evet



Şekil 6.13. Fedora Dağıtımında Kdenlive Uygulamasının Farklı Paket Formatlarında Disk Boyutu ve İşlem Performansı.

SONUÇLAR VE TARTIŞMA

Bu tez çalışmasında incelenen modern ve geleneksel paket formatlarının her birinin kendine özgü avantajları ve dezavantajları vardır. Özellikle modern paket biçimlerinin her birinin hala Linux dağıtımları için ana paket biçimi olarak uygulanabilir bir seçenek haline gelmesi için geliştirmeler yapılması gerekmektedir.

Linux dağıtımlarında son kullanıcının tek bir paket biçimine mahkûm olması bazen kısıtlamalara neden olmaktadır. İncelemesi yapılan modern paket biçimlerinin geleneksel olanlara alternatif olması son kullanıcının daha esnek çalışmasını sağlayacaktır.

Geleneksel paket biçimleriyle kıyasladığımızda AppImage, Flatpak ya da Snappy genel olarak bazı sorunlara sahipler ve uygulama çalıştırmak için daha fazla alan kullanmaktadırlar. Ancak bu formatların ayrı avantajları vardır. Özellikle Snappy ve Flatpak dağıtım içerisinde kendi sanal alanlarını oluştursalar da bu paket yöneticileri uygulamaların ihtiyaç duyduğu ortak kütüphaneler gibi dosyaları kendi arasında paylaştırabilmektedirler. AppImage, dağıtımlarda uygulama kullanmanın pratik bir yoludur. Ancak güncelleme işleminin burada son kullanıcılar açısından daha kullanışlı olmasını sağlamak için geliştirilmesi gerekmektedir.

Bu çalışmada incelenen paket yöneticilerinden Flatpak bir masaüstü oturumunda çalışacak şekilde tasarlanmıştır ve her ne kadar GUI'de iyi olsa da sistem üzerinde bazı masaüstü oturum servislerine bağlı olduğundan Linux kullanımının sunucularda yaygın olması açısından diğerlerine nazaran tam bir sunucu desteği bulunmamaktadır.

Geleneksel paket yöneticilerinde genellikle dağıtım üzerinde bulunan güncelleme araçları ile otomatik ya da manuel olarak paketlerin güncel sürümleri kontrol edilir ve güncellenirler. AppImage uygulamalarının yeni bir sürümünün güncellemesi ancak bir geliştirici tarafından paket yeniden oluşturulup İnternet'e servis edilirse yapılabilir. Güncellenen herhangi bir sürüm farklı bir AppImage uygulaması olarak sistemde çalıştırılır. Snappy paket yöneticisinin servisi paket güncellemelerini

normalde günde 4 kez denetler. Flatpak için ise CLI üzerinden komut vasıtasıyla paketlerin güncelleme işlemleri yapılabilir.

Diğer paket yöneticilerinin aksine sadece paket güncelleştirmelerini belli bir kanal üzerinden alamasa da AppImage aynı uygulamanın birden çok sürümünü kolaylıkla bir sistem üzerinde çalıştırabilir. Diğerlerinde bu avantajı oluşturabilmek için konteyner ve sanal makine monitörü gibi teknolojilerden faydalanılması ya da sanal alan ve farklı oturum gibi işlemler yapılması gereklidir.

AppImage dışındaki paket yöneticilerinin sisteme uygulama kurabilmesi için kök erişimi ya da süper kullanıcı ayrıcalığına ihtiyaç duyarken sadece AppImage buna ihtiyaç duymaz ve süper kullanıcı dışındaki kullanıcılar uygulama çalıştırabilirler.

Vlc uygulamasının yapılan karşılaştırma testlerinde Debian paketinin en az kurulum ve diskte alan boyutuna sahip olduğu görüldü. Bunun yanında AppImage formatının uygulamayı kararlı çalıştıramayıp sorunlara sahip olduğu gözlemlendi. Ayrıca AppImage dışındaki formatların uygulamayı açma ve çalıştırma performansları birbirine yakın ve hızlıydı.

Libreoffice uygulamasında Debian paketinin yine en az kurulum ve diskte alan boyutuna sahip olmasına rağmen dosya dönüştürme performansında AppImage ve Flakpak formatları daha iyiydi. Ayrıca Snappy dışındaki formatların uygulamayı açma ve çalıştırma performansları birbirine yakın ve hızlıydı.

Kdenlive uygulamasında Fedora dağıtımının paketi dosya dönüştürme performansında en iyi olandı. Burada da AppImage dışındaki formatların uygulamayı açma ve çalıştırma performansları birbirine yakın ve hızlıydı.

Modern formatlarda bir geliştirici, tüm bağımlılıkları içeren yazılım paketlerini oluşturacağından, paketteki bağımlılıkları en son güvenlik güncellemeleriyle güncel tutmak konusunda da sorumluluk sahibidir. Bir paket, yazılım geliştiricisi tarafından uygun şekilde korunmazsa, dağıtımdan bağımsız yazılım paketi sistem için güvenlik riski oluşturabilir. Bu aynı zamanda, dağıtım depoları aracılığıyla geliştiriciler tarafından bağımlılıkların korunduğu geleneksel yazılım paketlerine karşı ters bir durumdur.

KAYNAKLAR

Abate, P., DiCosmo, R., Treinen, R. & Zacchiroli, S. (2011). MPM: A Modular Package Manager. 14th International ACM Sigsoft Symposium on Component Based Software Engineering. DOI:10.1145/2000229.2000255

Ambler, S. (2002, Şubat 1). Some Useful Linux Commands. 25 Aralık 2019 tarihinde https://www.academia.edu/21146074/Some_useful_linux_commands adresinden erişildi.

AppImage. (2019). 25 Aralık 2019 tarihinde https://appimage.github.io adresinden erişildi.

Armasu, L. (2016, Haziran 22). Flatpak is the universal linux packaging format that puts security first. 25 Aralık 2019 tarihinde http://www.tomshardware.com/news/flatpack-universal-linux-packaging-format,32137.html adresinden erişildi.

Bazard, M. & Bhardwaj, S. (2011). Overview on Android – The New Mobile Operating System. SGI Reflections- International Journal of Science, *Technology and Management-Nisan*, *2*(1). http://sgi.ac.in/colleges/newsletters/1146090820110943321.pdf#page=27

Blokdijk, G. (2012). *Virtualization - The Complete Cornerstone Guide to Virtualization Best Practices: Concepts, Terms, and Techniques for Successfully Planning, Implementing and Managing Enterprise IT Virtualization Technology*(2. Baskı). London, UK:Emereo Pty Ltd

Borate, I. & Chavan, R.K. (2016). Sandboxing in Linux: From Smartphone to Cloud. *International Journal of Computer Applications-Ağustos*, *148*(8). DOI:10.5120/ijca2016911256

Brown, G.E., Eckhouse, R.H. & Estabrook, J. (1977). Operating System Enhancement through Firmware. Third International Computer Measurement Group Conference. DOI: 10.1145/1014196.803324

Catsoulis, J. (2009). An Introduction to Computer Architecture. Andy Oram(Ed.), *Designing Embedded Hardware*(2. Baskı) *içinde* (s. 13-57). Kaliforniya, ABD:O'Reilly Media.

Canonical. (2016). 25 Aralık 2019 tarihinde https://insights.ubuntu.com/2016/06/14/universal-snap-packages-launch-on-multiple-linux-distros/ adresinden erişildi.

Chroot. (2019, 27 Kasım). Gentoo Wiki içinde 25 Aralık 2019 tarihinde https://wiki.gentoo.org/wiki/Chroot adresinden erişildi.

CoreOS. (2019). 25 Aralık 2019 tarihinde https://coreos.com/rkt/ adresinden erişildi.

Courtès, L. (2013). Functional Package Management with Guix. European Lisp Symposium.

Dawson, M., DeWalt, B. & Cleveland, S. (2016). The Case for UBUNTU Linux Operating System Performance and Usability for Use in Higher Education in a Virtualized Environment. Southern Association for Information Systems (SAIS) 2016 Proceedıngs.

Dike, J. (2001). User-Mode Linux. 5th annual Linux Showcase & Conference.

DistroWatch. (2019). 25 Aralık 2019 tarihinde http://distrowatch.com/dwres.php?resource=popularity adresinden erişildi.

Dolstra, E., Löh, A. & Pierron, N. (2008). NixOS: A Purely Functional Linux Distribution. 13th International Conference on Functional Programming. DOI: 10.1017/S0956796810000195

Dumitraş, T., Dig, D. & Neamtiu, I. (2008). First ACM Workshop on Hot Topics in Software Upgrades (HotSWUp'08). OOPSLA Companion'08 Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications. DOI:10.1145/1449814.1449877

Dumitraş, T., Neamtiu, I. & Tilevich, E. (2009). Second ACM Workshop on Hot Topics in Software Upgrades (HotSWUp 2009). OOPSLA '09 Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications. DOI:10.1145/1639950.1639974

Dybvig, R.K. (1996). *The Scheme Programming Language: ANSI Scheme*(2. Baskı). New Jersey,ABD:Prentice Hall PTR Upper Saddle River

Fenzi, K. & Wreski, D. (2000). *Linux Security HOWTO*. Indiana, ABD:iUniverse

Flathub. (2019). 25 Aralık 2019 tarihinde https://flathub.org adresinden erişildi.

GNU Guix. (2019). 25 Aralık 2019 tarihinde https://guix.gnu.org/ adresinden erişildi.

GNU Project - Free Software Foundation. (1989). 25 Aralık 2019 tarihinde https://www.gnu.org/licenses/old-licenses/gpl-1.0.html adresinden erişildi.

Gonzalez-Barahona., J.M., Robles, G., Ortuno-Perez, M., Rodero-Merino, L., Centeno-Gonzalez, J., Matellán-Olivera, V., Castro-Barbero, E. & de-las-Heras-Quirós, P. (2005). Analyzing the Anatomy of GNU/Linux Distributions: Methodology and Case Studies (Red Hat and Debian). Stefan Koch (Ed.), *Free/Open Source Software Development içinde* (s. 27-58). Portland: Idea Group Inc (IGI).

Höbel, V. (2018). Flatpak, Snap ve AppImage. *ADMIN Network & Security Magazines-Ağustos, 45*.

Jha, S. (2019, Aralık 25). What is the purpose and use of software repository, package manager and source list in Linux based operating system. 25 Aralık 2019 tarihinde https://www.shaileshjha.com/what-is-the-purpose-and-use-of-repository-package-manager-and-source-list-in-linux-based-operating-system/ adresinden erişildi.

Kára, J. & Lihovarská, N. (2009). Ext 4 , btrfs , and the others. Linux Congress and Open Solaris Developers Conference.

Kim, T. & Zeldovich, N. (2010). Making Linux protection mechanisms egalitarian with UserFS. USENIX Security'10 Proceedings of the 19th USENIX conference on Security.

Koch, S. (2005). Evolution of Open Source Software Systems – A Large-Scale Investigation. First International Conference on Open Source Systems.

Krajci, I. & Cummings, D. (2013). History and Evolution of the Android OS. *Android on x86: An Introduction to Optimizing for Intel Architecture içinde* (s. 1-8). Berkeley, Kaliforniya:Apress

Kumaran, S.S. (2017). Introduction to Linux Containers. *Practical LXC and LXD : Linux Containers for Virtualization and Orchestration içinde* (s. 1-10). Berkeley, Kaliforniya:Apress

Lan, Y., Duan, X., Gao, J., Zhou, W. & Zhao, H. (2009). Extraction Methods on Linux Package Dependency Relations. International Conference on Information Engineering and Computer Science. DOI: 10.1109/ICIECS.2009.5362515.

Laurén, S., Rauti, S. & Leppänen, V. (2017). A Survey on Application Sandboxing Techniques. 18th International Conference on Computer Systems and Technologies. DOI:10.1145/3134302.3134312.

Li, Z., Kihl, M., Lu, Q. & Andersson, J.A. (2017). Performance Overhead Comparison between Hypervisor and Container Based Virtualization. 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA). DOI: 10.1109/AINA.2017.79.

Ljubuncic, I. (2019, Ağustos 29). A technical comparison between snaps and debs. 25 Aralık 2019 tarihinde https://snapcraft.io/blog/a-technical-comparison-between-snaps-and-debs adresinden erişildi.

Logix. (2018, Haziran 13). Winepak Is A Flatpak Repository For Windows (Wine) Games And Applications - Linux Uprising Blog. 25 Aralık 2019 tarihinde https://www.linuxuprising.com/2018/06/winepak-is-flatpak-repository-for.html adresinden erişildi.

MacKinnon, J.G. (1999). The Linux Operating System: Debian GNU/Linux. *Journal of Applied Econometrics*, *14*(4),443-452

Mancinelli, F., Boender, J., Di Cosmo, R., Vouillon, J., Durak, B., Xavier, L. & Treinen, R. (2006). Managing the Complexity of Large Free and Open Source Package-Based Software Distributions. 21st IEEE/ACM International Conference on Automated Software Engineering. DOI: 10.1109/ASE.2006.49.

Mongkolluksame, S., Issariyapat, C., Pongpaibool, P., Meesublak, K., Nulong, N. & Pukkawanna, S. (2012). A management system for software package distribution. 2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies.

Nandhini, U., Nivetha, B. & Shobana, D. (2016). An Analysis of Linux Operating System. *International Journal of Trend in Research and Development (IJTRD)-Şubat, 3*(1). http://www.ijtrd.com/papers/IJTRD1374.pdf

Nix. (2019). 25 Aralık 2019 tarihinde https://nixos.org/nix/ adresinden erişildi.

Norf, U. (2019). *The Role of Benchmarks in the Public Procurement of Computers*. Birleşik Krallık:Intel Corporation (UK) Ltd.

NSA. (2019). 25 Aralık 2019 tarihinde http://www.nsa.gov/selinux/ adresinden erişildi

EDITOR. (2019, Haziran 24). Linux Package Managers Compared – AppImage vs Snap vs Flatpak. 25 Aralık 2019 tarihinde https://www.ostechnix.com/linux-package-managers-compared-appimage-vs-snap-vs-flatpak/ adresinden erişildi.

Özdemir, H. (2018, Ocak 3). Linux Paket Yöneticileri. 25 Aralık 2019 tarihinde https://www.pythontr.com/makale/linux-paket-yoneticileri-632 adresinden erişildi.

Parrella, J.M. (2019). Package Management and Distribution in a Cloud World. Open Source Summit Japan and Automotive Linux Summit 2019.

Pavlov, A.I. & Cecchetti, M. (2008, Temmuz 24). SquashFS HOWTO. 25 Aralık 2019 tarihinde http://ldp.pakuni.net/HOWTO/pdf/SquashFS-HOWTO.pdf adresinden erişildi.

Salus, P.H. (1994). *A Quarter Century of UNIX*. New York, ABD:Addison-Wesley Publishing Co.

Similar projects. (2019, 15 Aralık). AppImageKit Wiki, GitHub içinde. 25 Aralık 2019 tarihinde https://github.com/AppImage/AppImageKit/wiki/Similar-projects adresinden erişildi.

Sivathanu, G., Wright, C.P. & Zadok, E. (2014). *Enhancing File System Integrity Through Checksums*(Technical Report FSL-04-04). New York, ABD: Computer Science Department, Stony Brook University.

Snapcraft. (2019). 25 Aralık 2019 tarihinde https://snapcraft.io adresinden erişildi.

Spinellis, D. (2012). Package Management Systems. *IEEE Software, 29*(2),84-86

Spinellis, D. (2017). A Repository of Unix History and Evolution. *Empirical Software Engineering-Haziran, 22*(3). DOI:10.1007/s10664-016-9445-5

Sueyasu, K., Tabata, T. & Sakurai, K. (2003). On the Security of Selinux with a Simplified Policy. Proc. of IASTED International Conference on Communication, Network, and Information Security (CNIS 2003).

Syed, M.H. & Fernandez, E.B. (2015). The Software Container Pattern. PLoP '15 Proceedings of the 22nd Conference on Pattern Languages of Programs.

Sysget. (2019). 25 Aralık 2019 tarihinde https://github.com/emilengler/sysget adresinden erişildi.

Tanenbaum, A.S. & Bos, H. (2014). *Modern Operating Systems*(4. Baskı). New Jersey,ABD:Prentice Hall PTR Upper Saddle River.

Tuna, G. (2008). Linux kümelerinin hata toleranslı uygulamalardaki önemi ve bir örnek uygulama(Yüksek lisans tezi), Erişim adresi: http://193.255.140.18/Tez/0070839/METIN.pdf.

Ubuntu. (2019). 25 Aralık 2019 tarihinde http://manpages.ubuntu.com/manpages/xenial/man1/alien.1p.html adresinden erişildi.

Verummeum. (2018). 25 Aralık 2019 tarihinde https://verummeum.com/portable-package-formats/ adresinden erişildi.

Viswanathan, A. & Neuman, B.C. (2013). A Survey of Isolation Techniques.

Wallen, J. (2018, Haziran 20). How to install Windows apps on Linux with Winepak - TechRepublic. 25 Aralık 2019 tarihinde https://www.techrepublic.com/article/how-to-install-windows-apps-on-linux-with-winepak/ adresinden erişildi.

Walters, C., Poo-Caamaño, G. & German, D.M. (2013). The future of continuous integration in GNOME. 1st International Workshop on Release Engineering (RELENG 2013). DOI: 10.1109/RELENG.2013.6607695.

Winepak. (2019). 25 Aralık 2019 tarihinde https://www.winepak.org/ adresinden erişildi.

Zimmerman, S. (2016, Ağustos 25). 25 Years of Linux and the GPL, and How Android Benefits. 25 Aralık 2019 tarihinde https://www.xda-developers.com/25-years-of-linux-and-the-gpl-and-how-android-benefits adresinden erişildi.

ÖZGEÇMİŞ

Adnan AKCAN, 1987 yılında Denizli'nin Çivril ilçesinde doğdu. İlkokulu Kıralan İlkokulu'nda ortaokulu Çıtak-Kıralan Ortaokulu'nda ve liseyi Çivril Emine Özcan Anadolu Lisesi'nde tamamladı. 2005 yılında lisans öğrenimi için başladığı Mersin Üniversitesi Bilgisayar Mühendisliği bölümünden 2009 yılında mezun oldu. 2011 yılında Edirne Trakya Tarımsal Araştırma Enstitüsü Müdürlüğü'nde bilgisayar mühendisi olarak çalışmaya başladı ve halen görevine devam etmektedir. 2018 yılında lisansüstü eğitimine Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda başladı ve halen eğitimine devam etmektedir.