Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma

Kandidaatintyö

Matias Turunen

RASPBERRY PI MINITIETOKONEEN MINIMAALINEN BOOTTAAVA KÄYTTÖJÄRJESTELMÄ

Työn tarkastaja(t): Apulaisprofessori Jussi Kasurinen

Työn ohjaaja(t): Apulaisprofessori Jussi Kasurinen

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma

Matias Turunen

Raspberry Pi minitietokoneen minimaalinen boottaava käyttöjärjestelmä

Kandidaatintyö

2020

26 sivua, 1 kuva

Työn tarkastajat: Apulaisprofessori Jussi Kasurinen

Hakusanat: käyttöjärjestelmä, Raspberry Pi
Keywords: operating system, Raspberry Pi

Tässä työssä tutkitaan oman minimalistisen käyttöjärjestelmän toteuttamista käytännössä. Työssä tarkastellaan muutamaa olemassa olevaa minimaalista käyttöjärjestelmää sekä käyttöjärjestelmiä yleisesti. Oman käyttöjärjestelmän toteutusta tutkitaan omatekoisen käyttöjärjestelmän avulla. Oman minimalistisen käyttöjärjestelmän ominaisuuksia tarkastellaan toteutuksessa ilmenneiden tarpeiden avulla. Työn aikana ei koskaan saatu valmiiksi omaa käyttöjärjestelmää, mutta opittiin paljon käyttöjärjestelmässä vaadituista ominaisuuksista sekä kuinka ensimmäisen oman käyttöjärjestelmän toteutusta olisi kannattanut lähestyä.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Program in Computer Science

Matias Turunen

Raspberry Pi microcomputer's minimal bootable operating system

Bachelor's Thesis

26 pages, 1 figures

Examiners: Assistant professor Jussi Kasurinen

Keywords: operating system, Raspberry Pi

This bachelor's thesis explores how to implement your own minimalistic operating system in practice. In this thesis we look at few existing minimal operating systems and operating systems in general. The implementation of proprietary operating system is studied using proprietary operating system. The features of own minimal operating system will be considered based on needs that have emerged in implementation. During the course of this thesis, I never completed my own operating system, but I learned a lot about the features required by the operating system and how approaching the implementation of my first operating system should have been done.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

ARM	Advanced RISC Machines
x86	Intelin kehittämä suoritinarkkitehtuuri
UNIX	Laitteistoriippumaton käyttöjärjestelmä
MIT	Massachusetts Institute of Technology
LAN	Local Area Network (Paikallisverkko)
USB	Universal Serial Bus
GPIO	General Purpose Input Output
UART	Sarjaliikenneprotokolla

1 JOHDANTO

Tässä luvussa käsitellään lyhyesti työn taustoja, asetetaan työlle tavoitteet sekä rajaukset, esitellään työn ratkaisumenetelmät ja käydään läpi työn rakenne.

1.1 Tausta

Erilaiset käyttöjärjestelmät ovat olleet kiinnostavia jo pidemmän aikaa. Näihin voidaan lukea Windows, MacOS sekä erilaiset Linuxin variaatiot. Kaikille käyttöjärjestelmille on omat tarkoituksensa ja käyttäjäkuntansa. Tunnetuimmat käyttöjärjestelmät ovat laajoja kokonaisuuksia, joissa on paljon ominaisuuksia. Tavallinen käyttäjä ei välttämättä koskaan tarvitse niistä kuin osan. Minimaaliset käyttöjärjestelmät on tehty yhteen käyttötarkoitukseen kerrallaan, esimerkiksi verkkoselaimen käyttöön tai tekstinkäsittelyyn. Tässä työssä käsitellään oman minimalistisen käyttöjärjestelmän toteutusta.

1.2 Tavoitteet ja rajaukset

Tässä kandidaatintyössä tavoitteena on selvittää mikä on mahdollisimman pieni toteutettavissa oleva käyttöjärjestelmä. Tarkemmin sanottuna työssä tutkitaan, mitä osia vaaditaan pienimpään mahdolliseen boottaavaan käyttöjärjestelmään, mitä toiminnollisuuksia pienimmässä mahdollisessa käyttöjärjestelmässä tulisi olla ja kuinka käyttöjärjestelmän toteutusta kannattaa lähestyä.

Työssä keskitytään erityisesti ARM-pohjaisiin (Advanced RISC Machines) käyttöjärjestelmiin ja Raspberry Pi -minitietokoneeseen, mutta esimerkkejä voidaan tuoda myös muista arkkitehtuureista, esimerkiksi x86 (Intelin kehittämä suoritinarkkitehtuuri) arkkitehtuurista. Lisäksi työssä keskitytään ei-graafisiin käyttöjärjestelmiin. Tämän työn ohessa on tarkoitus toteuttaa oma minimalistinen käyttöjärjestelmä, jossa on vain perustoimintoja.

1.3 Työn rakenne

Ensimmäisessä luvussa käydään läpi työn taustat, rajaukset sekä työn rakenne. Toisessa luvussa tarkastellaan käyttöjärjestelmiä yleisesti. Kolmannessa luvussa tarkastellaan kahta

olemassa olevaa minimaalista käyttöjärjestelmää. Molemmista käyttöjärjestelmistä käydään läpi tärkeimmät ominaisuudet sekä niiden suunniteltu käyttötarkoitus. Neljännessä luvussa kerrotaan ARM arkkitehtuurista ja Raspberry Pi minitietokoneesta. Viidennessä luvussa käsitellään työn tuloksia. Lisäksi luvussa kerrotaan oman, työssä suunnitellun minimalistisen käyttöjärjestelmän osista. Kuudennessa luvussa pohditaan saatuja tuloksia ja sitä, kuinka tuloksia voitaisiin hyödyntää tulevaisuudessa. Viimeisessä eli seitsemännessä luvussa on vielä yhteenveto.

2 KÄYTTÖJÄRJESTELMISTÄ YLEISESTI

2.1 Käyttöjärjestelmän yleisimmät ominaisuudet

Käyttöjärjestelmissä on monia ominaisuuksia. Jotkut ominaisuuksista ovat riippuvaisia käyttöjärjestelmän tyypistä, eli siitä kuinka käyttöjärjestelmän ydin on rakennettu. Monoliittisessa ydintyypissä koko käyttöjärjestelmää suoritetaan yhtenä ohjelmana ytimen kanssa samassa tilassa. Toinen käyttöjärjestelmätyyppi on kerroksittainen systeemi. Kerroksittaisessa systeemissä käyttöjärjestelmän suoritus on jaettu kerroksiin, joista jokaisella on oma tehtävänsä, kuten muistinhallinta, prosessien hallinta tai I/O hallinta. Kolmas ydintyyppi on mikroydin, jossa vain pieni osa koodia ajetaan ydintilassa ja muu koodi suoritetaan tilassa, jossa on vähemmän oikeuksia. Hyötynä tässä tavassa on se, että ydinbugien määrää saadaan pienennettyä, mikä puolestaan vähentää vakavien virheiden määrää käyttöjärjestelmässä. (Tanenbaum, 2001)

Yleisiä, lähes kaikissa käyttöjärjestelmissä olevia ominaisuuksia ovat prosessit, tiedostot ja kansiot, I/O laitteet, tiedostojen suojaus ja komentokehote (engl. shell). (Tanenbaum, 2001)

2.1.1 Prosessit

Prosessit ovat käyttöjärjestelmän ydinkomponentteja. Käytännössä prosessi on suoritettava ohjelma, joka tekee jonkin asian. Käyttöjärjestelmässä voidaan suorittaa useita prosesseja näennäisesti yhtäaikaisesti, jolloin yhtä prosessia suoritetaan ja muut prosessit odottavat, että prosessori vapautuu niiden käyttöön. Prosessi voi pysähtyä kesken suorituksen ja jatkaa myöhemmin. Tällöin prosessin kaikki tiedot tallennetaan muistiin, jotta suoritusta voidaan jatkaa täsmälleen samasta tilasta, jossa prosessi oli ennen keskeytystä. (Tanenbaum, 2001)

Prosessit voivat luoda uusia prosesseja ja prosessit voivat keskustella keskenään käyttämällä prosessien välistä kommunikaatiota. Kommunikaatiota on kahta tyyppiä, muistin jakaminen ja viestien lähetys. Muistinjaossa prosesseilla on pääsy samaan muistialueeseen ja tieto siirtyy, kun ne lukevat ja kirjoittavat tälle muistialueelle. Viestien välityksessä prosessit lähettävät toisilleen viestejä. Viestien välitys soveltuu paremmin

pienille datamäärille, kun taas muistin jako on parempi suurille määrille dataa. Toisaalta viestien välitys on hitaampaa kuin muistin jakaminen. (Silberschatz, 2012)

Käyttöjärjestelmässä prosesseja hallinnoi prosessien hallinnoija, joka päättää mitä prosessia suoritetaan milloinkin, ja pitää kirjaa siitä mitä tietoa milläkin prosessilla on käytettävissään. Monissa käyttöjärjestelmissä on niin kutsuttu prosessitaulu, jonne talletetaan tiedot jokaisesta sillä hetkellä olevasta prosessista. Käyttöjärjestelmä voi lähettää prosessille hälytyssignaalin, joka pakottaa prosessin keskeyttämään sen, mitä se oli tekemässä, tallentamaan tietonsa rekistereihin ja alkaa suorittaa signaalinkäsittely toimintoa. Kun signaali on käsitelty, prosessi saa jatkaa suoritustaan normaalisti. (Tanenbaum, 2001; Silberschatz, 2012)

2.1.2 Tiedostojärjestelmä

Toinen lähes kaikissa käyttöjärjestelmissä oleva perusominaisuus on tiedostojärjestelmä. Sen tehtävänä on piilottaa käyttäjältä levy- ja I/O -toiminnot ja tarjota helppokäyttöinen järjestelmä tiedostojen käsittelyyn. Tietysti käyttäjä tarvitsee systeemikäskyjä tiedostojärjestelmän käyttöön, mutta hänen ei tarvitse tietää alla olevaa levy- ja I/O toimintojen pinoa. Useissa käyttöjärjestelmissä tiedostojen lajitteluun käytetään kansioita. Systeemikäskyjä tarvitaan tiedostojen lisäämiseksi kansioihin. Kansiossa voi olla sekä tiedostoja että muita kansioita, jolloin kansiorakenteesta muodostuu puumainen. (Tanenbaum, 2001)

Prosessilla on aina työkansio. Prosessi voi vaihtaa työkansiotaan systeemikäskyn kautta. Aina ennen kuin tiedostoa voi lukea tai kirjoittaa, on tiedosto avattava. Tiedostoa avatessa tarkistetaan, onko käyttäjällä oikeus kyseiseen tiedostoon, jos kyllä niin palautetaan tiedostopointteri, jos ei niin palautetaan virhe. (Tanenbaum, 2001; Rosenblum, 1992)

2.1.3 I/O laitteet ja tiedostojen suojaus

Käyttöjärjestelmän tulee kontrolloida erilaisia I/O-laitteita. Näitä ovat esimerkiksi näytöt, näppäimistöt, hiiret, tulostimet jne. Käyttöjärjestelmässä tulee olla ajurit eri laitteille, eli käyttöjärjestelmän tulee osata käyttää kytkettäviä laitteita.

Useimmissa käyttöjärjestelmissä tiedostoilla on käyttöoikeudet, jotka määrittävät kuka saa suorittaa, lukea tai kirjoittaa tiedostoa. Käyttöoikeudet on määrätty ennalta ja niihin voi tehdä muutoksia vain tiedoston omistaja tai pääkäyttäjä. (Tanenbaum, 2001)

2.1.4 Komentokehote

Komentokehote on prosessi, joka käynnistetään heti käyttöjärjestelmän käynnistymisen jälkeen. Komentokehote ei kuitenkaan ole osa itse käyttöjärjestelmää. Käyttäjä voi syöttää komentokehotteeseen komentoja, jotka se suorittaa. Komentokehote hoitaa komentojen putkitukset ja uudelleenohjauksen. Komentokehotteen tilalla voi olla myös graafinen käyttöliittymä. (Tanenbaum, 2001)

3 OLEMASSA OLEVIA MINIMAALISIA KÄYTTÖJÄRJESTELMIÄ

3.1 XV6

XV6 on minimaalinen opetuskäyttöön suunnattu käyttöjärjestelmä, jonka ovat tehneet Russ Cox, Frans Kaashoek ja Robert Morris. XV6:ssa on toteutettu kaikki oleelliset osat, jotka kuuluvat käyttöjärjestelmään. Se perustuu Dennis Ritchien ja Ken Thompsonin Unix version 6 (v6) käyttöjärjestelmään ja on täten x86-pohjainen käyttöjärjestelmä. XV6:a on käytetty käyttöjärjestelmien opetukseen mm. MIT:ssä (Massachusetts Institute of Technology). (Cox, 2014)

XV6:ssa on oma tiedostojärjestelmä, prosessien ajoitus, lukitusjärjestelmä sekä poikkeustenhallinta. Tiedostojärjestelmä on Unix-tyylinen ja koostuu tiedostoista ja kansioista. Tiedostojärjestelmässä on seitsemän tasoa (järjestyksessä): ”tiedoston kuvaus”, ”polun nimi”, ”kansio”, ”inode”, ”lokitus”, ”puskurin välimuisti” sekä ”kiintolevy”. Prosessin ajoitus on tehty käyttäen round-robin-periaatetta, eli jokaista prosessia ajetaan vuorotellen. Tämä tapa ei ole ideaalinen, mutta monimutkaisemman prioriteetteihin perustuvan menetelmän tekeminen ei ole järkevää näin pienessä käyttöjärjestelmässä. Lukitusjärjestelmä on käytössä XV6:n ydintoiminnoissa ja se käyttää enintään kahden lukon ketjuja. Ohjelmissa, joita voidaan ajaa XV6-ympäristössä ei tarvita lukkoja, koska kaikki prosessit toimivat yhdessä säikeessä ja ne eivät jaa muistia. (Cox, 2014)

XV6:n käynnistyslatain (engl. bootloader) on vain 470 bitin kokoinen. Suurin sallittu x86 käynnistyslatain on 512 bittiä. Jotta käynnistyslataimesta on saatu näin pieni, oletetaan että käyttöjärjestelmän ydin on käynnistyslevyllä yhtenäisenä koodina osiossa yksi. Käynnistyslatain lataa ytimen ja alkaa suorittaa sitä. Monimutkaisemmissa järjestelmissä käynnistys tapahtuu kahdessa vaiheessa: yksinkertainen käynnistyslatain lataa monimutkaisemman (ja suuremman) käynnistimen levyltä. Lopulta tämä monimutkaisempi käynnistyslatain lataa ytimen ja alkaa suorittaa sitä. (Cox, 2014)

3.2 PicOs

PicOs on erittäin pieniin ympäristöihin suunniteltu käyttöjärjestelmä. Oikeammin sanottuna se on oikeastaan vain kokoelma funktioita, ja tätä kokonaisuutta voidaan kutsua käyttöjärjestelmäksi. Se on tarkoitettu pieniin mikrokontrollerilaitteisiin, joissa on vain vähän muistia, laitteet ovat matalakustanteisia, ne ovat erittäin kestäviä ja niillä on matala virrankulutus. Laitteet voivat keskustella keskenään pienellä alueella käyttämällä niiden omia lähettimiä. (Akhmetsina, 2003)

Multiprosessointi PicOs-käyttöjärjestelmässä ei ole tavallista. Eri tehtävät jakavat saman globaalin pinon ja toimivat samanaikaisina rutiineina, joilla on useita syötepisteitä ja implisiittinen ohjauksen siirto. Jokainen tehtävä on äärellinen tilakone, joka vaihtaa tilaa vastauksena tapahtumiin. PicOs:in tehtäviä voidaan kutsua myös prosesseiksi. Jokaisella prosessilla on uniikki tunniste, joka annetaan sille prosessia luotaessa. Prosessit voivat pysähtyä odottamaan ja jatkaa toimintaansa myöhemmin. Kun prosessi jatkaa toimintaansa, kaikki sille jonossa olleet tapahtumat ohitetaan, eli niitä ei käsitellä. Siitä hetkestä, kun prosessi jatkaa toimintaansa, se ei voi menettää käyttöön saamaansa prosessoria, ellei se menetä sitä ytimelle. (Akhmetsina, 2003)

PicOs:n vaatima muistimäärä on erittäin pieni. Se tarvitsee vain ytimen ja prosessien viemän tilan, eli pinon koko riippuu näistä kahdesta. PicOs:n ensimmäisessä versiossa pinon koko on 512 bittiä, ja tämä näyttää olevan enemmän kuin tarpeeksi ohjelmille, joita suoritetaan PicOs ympäristössä. PicOs-käyttöjärjestelmässä on useita ajureita eri laitteille. Siinä on sarjaportteja, LCD-näyttö ja LAN (Internet) liitäntä. Vaikka kaikki edellä mainitut palvelut olisivat käytössä samanaikaisesti, ei PicOs veisi enempää kuin 4Kb muistia. Käyttöjärjestelmässä on muistinhallinta, joka pystyy jakamaan keon useisiin erottamattomiin pooleihin. Kun muistia on vähän, tällainen pooleihin perustuva järjestelmä on parempi kuin yksi suuri pooli, koska eri sovellukset pystyvät vastaamaan virheisiin eri tavalla ja kriittiset komponentit ovat itsenäisempiä. (Akhmetsina, 2003)

PicOs:aa on suunnitelmissa kehittää edelleen ja lisätä siihen ominaisuus, joka sallii prosessien ryhmittelyn joukkoihin. Joukkojen sisällä prosessit käyttäytyvät samoin kuin ennenkin, mutta joukkojen toiminnot olisivat ennakoitavissa. Eri joukkoihin kuuluvat

prosessit kommunikoisivat keskenään käyttämällä tarkoitukseen suunniteltuja signaalijonoja. PicOs:n yksi hyvä ominaisuus onkin, että säikeiden synkronointi ongelmaa ei ole. Lisäksi PicOs prosessit ovat yleisesti uudelleenkäytettävämpiä ja itsedokumentoivia. (Akhmetsina, 2003)

4 ARM PROSESSORIPERHE JA RASPBERRY PI

4.1 ARM:n historiaa

Englannissa vuonna 1983 perustettiin Acorn Computers ltd, joka myöhemmin samana vuonna aloitti Acorn RISC Management (ARM) projektin, jonka tavoitteena oli suunnitella yhtiön oma prosessori. Ensimmäinen ARM-prosessori julkaistiin vuonna 1985 ja sitä kutsuttiin nimellä ARM1. Kun ARM2:ta alettiin kehittää, tuli Apple mukaan suunnitteluun, sillä se halusi käyttää ARM:ia omiin tarpeisiinsa, mutta koska ARM ei vielä silloin sisältänyt yhtenäistä muistinhallintayksikköä, tarvittiin Applen apua. Tämän yhteistyön seurauksena syntyi ARM Ltd vuonna 1990. Tämän jälkeen ARM lyhenne muuttui tarkoittamaan Advanced RISC Machinea. (Hariprasad, 2017)

4.2 ARM:n ominaisuuksia

ARM arkkitehtuuri on yksinkertainen, mikä mahdollistaa toimintojen jättämisen pois itse mikrosirulta, joka taas mahdollistaa entistä pienempien ja edullisempien sirujen valmistuksen. ARM prosessoreissa on Thumb käskykanta, joka tiivistää 32 bittisen koodin 16 bittiseksi. Tämä mahdollistaa entistä pienempien ja tehokkaampien ohjelmien kirjoittamisen. ARM oli pitkään 32 bittinen järjestelmä, mutta 64 bittisyys tuli mahdolliseksi ARMv8-A arkkitehtuurin myötä. (ARM, n.d) ARM ytimet ovat yksinkertaisia verrattuna muihin prosessoreihin, sillä ne voidaan valmistaa vain muutamista transistoreista. Nykyään ARM prosessoreita on käytössä puhelimissa, televisioissa ja kameroissa. Myös Raspberry Pi: ssä on ARM prosessori. (Tung, 2018; Hariprasad, 2017)

ARM prosessorissa on 31 yleiskäyttöistä 32 bittistä rekisteriä, joista 16 on käyttäjän muokattavissa. Rekistereistä kaksi on tärkeässä asemassa. Toinen rekistereistä (CPSR) tallettaa nykyisen ohjelman statuksen ja toinen (SPSR) on tallennetun ohjelman rekisteri. Poikkeustilanteessa CPSR-rekisterin tiedot kopioidaan SPSR-rekisteriin. (Hariprasad, 2017)

ARM prosessoreita on tehty moniin erilaisiin käyttötarkoituksiin. ARM prosessoriperheestä löytyy sekä vähävirtaisia, pieniin älylaitteisiin sopivia prosessoreja,

kuten myös tehokkaampia laskentakäyttöön tarkoitettuja malleja. Nykyään ARM prosessoreja on viidessä eri käyttötarkoitusluokassa. Cortex-A prosessorit ovat kaikkein tehokkaimpia ja tarkoitettu laskentaan teollisuudessa ja tiedon varastoinnissa. Cortex-R on seuraavaksi tehokkain malli, ja käyttötarkoitukset vaihtelevat kameroista lääketieteeseen. Cortex-M on pienin tehoiltaan ja sitä käytetään puettavissa laitteissa sekä sensoreissa. ARM koneoppimiseen optimoituja prosessoreja käytetään tekoälyissä, neuroverkoissa ja virtuaalitodellisuudessa. Viimeinen tuoteryhmä on turvalliset prosessorit, joita käytetään sähköisissä passeissa, maksukorteissa ja SIM-korteissa. (ARM, n.d)

4.3 Raspberry Pi

Raspberry Pi on noin luottokortin kokoinen minitietokone. Uusimmassa Model 3B+ versiossa on ARMv8 arkkitehtuurin Cortex-A53 neliydinprosessori ja 1 Gb keskusmuistia. Tietokoneessa on myös bluetooth, langaton verkkoyhteys sekä kiinteä verkkoyhteys. Lisäksi laitteesta löytyy liitäntöjä näytölle, USB-oheislaitteille sekä kamera- ja muille moduuleille. Oleellisena osana Raspberry Pi tietokoneeseen kuuluu myös GPIO (General purpose Input Output) -pinnejä, joita voi käyttää ohjelmoinnissa. GPIO -pinneihin voi kiinnittää esimerkiksi näyttöjä ja erilaisia sensoreita. (Raspberry Pi Foundation, n.d; Tung, 2018)

5 TULOKSET

5.1 Ohjelmatoteutus

Työssä lähestyttiin minimalistisen käyttöjärjestelmän ajatusta yrittämällä kirjoittaa oma minimalistinen käyttöjärjestelmä Raspberry Pi 3b+ minitietokoneelle. Ohjelmaa tehtiin jo olemassa olevien toteutusten pohjalta, sekä seurattiin yhtä internetissä saatavilla olevaa ohjetta, jonka tuloksena olisi ollut minimalistinen käyttöjärjestelmä Raspberry Pi 2 minitietokoneelle. Koska kohdetietokoneen versio 3 ei juurikaan poikkea versiosta 2, todettiin että version 2 ohjeilla päästäisiin toteutuksessa hyvin alkuun. Versioiden 2 ja 3 suurin ero on, että versiossa 3 on tehokkaampi suoritin ja enemmän keskusmuistia.

Ohjelman testaamista varten käytössä oli ensin Qemu emulaattorin versio 2.81, jossa ei ole suoraa tukea Raspberry Pi minitietokoneen kolmannelle versiolle. Myöhemmin Qemu versio vaihtui versioon 4.0.1 joka tukee suoraan Raspberry Pi versiota kolme. Aiemmalla versiolla päästiin kuitenkin kehitystyössä hyvin alkuun. Lisäksi käytettävissä oli oikea Raspberry Pi 3b+ minitietokone, tosin ohjelmaa ei koskaan kunnolla päästy suorittamaan laitteella johtuen kohdatuista haasteista, jotka kuvataan myöhemmin. Ohjelman toiminta emulaattorissa ennen sen muuntamista 64-bittikseksi on nähtävillä kuvasta 1. Toistaiseksi tuntemattomasta syystä valkoinen näkyy emulaattorissa suoritetussa ohjelmassa punaisena. Tätä värivirhettä ei pystytty jäljittämään ohjelmakoodiin, sillä tekstin väriarvojen muutoksella ei ollut mitään vaikutusta tekstin väriin. Värejä olisi ollut tarkoitus kokeilla uudestaan 64-bittisellä versiolla, mutta sinne asti ei koskaan päästy. Käyttöjärjestelmän nimi on ”Matix”, nimetty tekijänsä mukaan.

Kuva 1. Minimalistisen käyttöjärjestelmän suoritusta Qemu emulaattorissa.

Raspberry Pi 3b+ minitietokoneessa on ARMv8 arkkitehtuurin suoritin, jota voidaan käyttää 32 bittisessä tilassa. (ARM, n.d) Tämän ominaisuuden tulisi mahdollistaa 32 bittisen koodin suorittaminen 64 bittisyyden sijaan. Käyttöjärjestelmästä päätettiin tehdä 32 bittinen, koska sen oletettiin olevan helpompaa toteuttaa kuin 64 bittinen versio. Myöhemmin siirryttiin kuitenkin 64 bittiseen järjestelmään, kun havaittiin että 32 bittinen koodi ei toimi fyysisessä laitteessa. Haastavinta oli löytää 64 bittinen käynnistyskoodi, sillä sellaisen tekemisen opetteluun ei haluttu käyttää aikaa tässä työssä, eikä se olisi ollut työn luonteen kannalta järkevää. Ohjelman kääntämiseksi kohdeympäristöön sopivaksi käytettiin aarch64-elf kääntäjää.

Ohjelmatoteutuksessa ei koskaan tullut valmista käyttöjärjestelmää, jossa voisi suorittaa yhtä tai useampaa ohjelmaa. Toteutuksesta päätettiin luopua, kun oikealle näytölle piirtämistä koskevassa vaiheessa havaittiin, että kuvapuskurin (engl. framebuffer) osoitetta haettaessa postilaatikosta (engl. mailbox) kuvapuskurin osoitteeksi tulee 0, vaikka näin ei pitäisi tapahtua. Ongelmaa yritettiin korjata pienillä muutoksilla tuloksetta, jolloin

päädyttiin tulokseen, että yli 60% koko ohjelmasta (yli 1300 riviä) pitäisi kirjoittaa uudestaan. Tähän uudelleen kirjoitettavien osien joukkoon kuuluvat mm. kuvapuskuri, postilaatikko sekä muistinhallinta, joiden toiminnasta kerrotaan myöhemmässä luvussa. Ohjelmassa toteutetut käyttöjärjestelmän osat ovat: UART (sarjaliikenneprotokolla), muistinhallinta, kuvapuskuri, postilaatikko sekä osa standardi kirjastoa.

Ohjelmatoteutuksessa toimivaksi saatiin muistinhallinta sekä näytölle piirtäminen. Näytölle piirto vaatii muistin hallintaa, UART: n, kuvapuskurin ja postilaatikon. Näytölle piirtäminen testattiin toimivaksi emulaattorissa, jolloin käytössä oli vielä emulaattorin versio Raspberry Pi 2 tietokonetta varten ja käyttöjärjestelmä oli 32 bittinen. Tällöin käyttöjärjestelmässä oli käytössä demo-ohjelma, joka tulosti näytölle jokaisen kirjoitetun merkin. Kuitenkin kun siirryttiin 64 bittiseen käyttöjärjestelmään ja Qemu versioon, joka tukee Raspberry Pi 3b+ tietokonetta, ei näytölle piirto enää toiminut edellä kuvatusta syystä.

5.2 Minimalistisen käyttöjärjestelmän osia

Raspberry pi 3b+ minitietokoneelle tarkoitetussa minimalistisessa käyttöjärjestelmässä tulee olla vähintään seuraavat ominaisuudet: UART, muistinhallinta, kuvapuskuri, postilaatikko, oma standardikirjasto, jonkinlainen toteutus näytölle piirrettävistä kirjainmerkeistä, keskeytykset, prosessit, lukot ja poikkeukset.

5.2.1 UART

UART vastaa sarjaliikenteestä Raspberry Pi minitietokoneessa. Sen toimintoihin kuuluu merkkien vastaanottaminen sarjaporttiin liitetyltä laitteelta kuten näppäimistöltä. Lisäksi sen kautta voidaan kirjoittaa sarjaväylään ja näin lähettää komentoja sarjaväylässä oleville laitteille.

5.2.2 Muistinhallinta

Muistinhallinnan tehtäviin kuuluu pitää kirjaa muistissa olevista alueista. Tietokoneen muisti on jaettu useisiin ennalta määrätyn kokoisiin alueisiin. Muisti alustetaan jakamalla

käytettävissä oleva muisti keoksi ja käyttömuistiksi. Ohjelmat voivat varata itselleen muistia käyttömuistista, kun keko on varattu käyttöjärjestelmälle. Käyttömuistin alustus tapahtuu luomalla linkitetty lista käytettävissä olevista muistipalasista. Ohjelmat voivat sitten pyytää itselleen osia tästä alueesta. Kun ohjelma ei enää tarvitse muistia, se antaa luvan vapauttaa muistin ja tällöin muistinohjain liittää käytössä olleet muistipalat takaisin osaksi vapaiden alueiden listaa.

5.2.3 Kuvapuskuri

Kuvapuskuri on ennalta määrätyn kokoinen alue muistissa, jonka sisältö voidaan näyttää näytöllä. Raspberry Pi tietokoneessa kuvapuskurin osoitteen saa pyytämällä sen postilaatikosta. Kuvapuskurin muistialueelle voidaan kirjoittaa bittejä, joiden arvon mukaan voidaan muuttaa näytöllä näkyvää kuvaa. Kuvapuskurissa voi olla kerralla vain yksi kuva, joten esimerkiksi videon näyttämiseksi puskurissa olevaa tietoa pitää jatkuvasti päivittää.

5.2.4 Postilaatikko

Raspberry Pi minitietokoneessa postilaatikon välityksellä keskustellaan prosessorin ja näytönohjaimen välillä. Postilaatikon kautta voidaan esimerkiksi pyytää kuvapuskurin osoite, jos prosessorilla suoritettavassa ohjelmassa halutaan piirtää näytölle. Näytölle piirtäminen tapahtuu kuvapuskurin kautta. Muita postilaatikon kautta tehtäviä toimintoja ovat mm. virranhallinta, painikkeiden toiminta sekä laitteessa olevien valojen komentaminen.

5.2.5 Standardikirjasto

Standardikirjaston tehtävänä on sisältää käyttöjärjestelmän tarvitsemia toimintoja. Standardikirjaston toiminnot eivät ole riippuvaisia mistään muista toiminnoista, vaan ne toteuttavat toiminnallisuutensa ilman minkään muun toiminnon apua. Standardikirjaston toimintoja ovat mm. tekstimuotoisen numeron muuttaminen numeromuotoiseksi numeroksi, numeron muuttaminen tekstiksi, tietueen nollaus, muistin kopiointi sekä muita usein tarvittuja toimintoja.

5.2.6 Fontit

Jokainen käyttöjärjestelmä tarvitsee oman toteutuksensa näyttölaitteelle piirrettävistä kirjainmerkeistä. Kirjainmerkit ovat bittikarttoja, jotka on määritelty suoraan käyttöjärjestelmän ytimen koodiin. Ei ole määritelty miltä kirjainmerkkien tulee näyttää, joten toteutuksia voi olla yhtä monta kuin on erilaisia käyttöjärjestelmiäkin. Käyttöjärjestelmän kirjainmerkeissä usein on määriteltynä yleisimmät kirjainmerkit, aakkoset sekä isoin että pienin kirjaimin, numerot ja erikoismerkit. Lisäksi määriteltynä on jokin merkki tapauksille, joissa järjestelmästä ei löydy sopivaa merkkiä.

5.2.7 Keskeytykset

Keskeytys nimensä mukaisesti keskeyttää parhaillaan suoritettavan ohjelman suorituksen ja siirtyy suorittamaan keskeytyksen tyypille sopivaa koodia. Keskeytys voi tulla missä kohtaa tahansa ohjelman suoritusta, eikä sitä voi välttää. Keskeytys voi johtua ohjelman virheestä tai laitteistosta. Laitteistokeskeytyksen tapauksessa laitteistossa, esim. levyasemassa tapahtuu virhe, jonka välitön käsittely vaatii suoritettavan ohjelman keskeyttämistä.

5.2.8 Prosessit

Prosessit ovat suoritettavia ohjelmia tai osia ohjelmasta, joka koostuu useista prosesseista. Prosessilla on kolme mahdollista tilaa, joissa se voi olla suorituksensa aikana. Prosessi voi olla suorituksessa, jolloin siihen liittyvää koodia suoritetaan aktiivisesti suorittimessa. Toinen vaihtoehto on, että prosessi on odottavassa tilassa. Se voi odottaa joko laitteiston vastausta tai toista prosessia. Kolmas mahdollinen prosessin tila on olla valmiustilassa, joka tarkoittaa, että prosessi on valmiina alkamaan tai jatkamaan suoritustaan.

5.2.9 Lukot

Lukkoja tarvitaan, kun useat prosessit haluavat käyttää samoja laitteisto- tai tiedostoresursseja. Prosessi voi pyytää tiedoston tai laitteen lukitsemista niin, että vain lukitusta pyytäneellä prosessilla on pääsy lukituksen kohteena olevaan laitteeseen tai tiedostoon. Tyypillisintä on lukita tiedosto siihen kirjoittamisen ajaksi, jottei toinen

prosessi lue siitä keskeneräistä ja tällöin mahdollisesti virheellistä tietoa. Lisäksi se estää kahta prosessia kirjoittamasta samaan tiedostoon yhtä aikaa, jotta tiedon eheys säilyy.

5.2.10 Poikkeukset

Poikkeukset johtuvat ohjelman suorituksen virhetilanteesta, joka voi olla riippuvainen laitteistosta tai ohjelmointikielen rajoituksista. Laitteiston riippuvia poikkeuksia ovat mm. nollalla jakaminen, virheellinen prosessorille lähetettävä komento tai viittaus muistialueeseen, joka ei ole olemassa tai on olemassa prosessille varattujen alueiden ulkopuolella. Ohjelmointikielestä johtuvat rajoitukset voivat olla esimerkiksi tekstin jakolasku tai väärän tietotyypin tallentaminen toiselle tyypille varatulle muistipaikalle. Poikkeukset johtavat ohjelman suorituksen keskeytymiseen, ellei poikkeusta ole hallittu ohjelman sisällä.

6 POHDINTA JA TULEVAISUUS

Tätä työtä tehtäessä havaittiin, että oman käyttöjärjestelmän tekeminen on ainakin teoriassa mahdollista. Omaa käyttöjärjestelmää suunniteltaessa on otettava huomioon laitteiston ja käytettävissä olevan prosessoriarkkitehtuurin tuomat rajoitukset ja mahdollisuudet. Tässä työssä oli aluksi tarkoituksena tehdä 32-bittinen käyttöjärjestelmä Raspberry Pi minitietokoneelle, jonka tuotetiedoissa lukee, että laite tukee sekä 32 että 64-bittisiä käyttöjärjestelmiä. Toteutuksen edetessä laitteistotestaukseen huomattiin kuitenkin, että 32-bittinen koodi ei toimi itse laitteessa. Tietoa siitä, oliko vika sitten itse laitetiedoissa vaiko lukijan tulkinnassa ei tarkemmin tutkittu. Johtopäätöksenä tästä saadaan, että kaikkiin valmistajan antamiin tietoihin ei välttämättä kannata luottaa, varsinkin kun kyseessä on uusi teknologia, laite tai molemmat.

Minimaalisen käyttöjärjestelmän voisi myös toteuttaa käyttäen pohjana jotakin jo olemassa olevaa pientä käyttöjärjestelmää, joka on testattu toimivaksi kohdeympäristössä. Tästä pohjasta sitten poistetaan ominaisuuksia, joita ei koeta tarpeelliseksi omassa käyttöjärjestelmässä. Useat Linux pohjaiset käyttöjärjestelmät ovat avointa lähdekoodia, ja niiden muokkaaminen on sallittua. Lisäksi on todennäköistä, että valmiissa käyttöjärjestelmässä perusasiat on jo toteutettu tehokkaimmalla mahdollisella tavalla, jolloin ei tarvitse ns. keksiä pyörää uudestaan.

Tulevaisuudessa suunniteltaessa ensimmäistä omaa käyttöjärjestelmää kannattaa valita kohteeksi hyvin tunnettu ja yleinen laite ja arkkitehtuuri. Uusimmille laitteille ja arkkitehtuureille ei niiden elinkaaren alkuvaiheessa löydy vielä suurta määrää valmiita projekteja, keskusteluita ja rakennusohjeita toisin kuin jo muutaman vuoden olemassa olleelle laitteelle. Tässä työssä valittiin uusi 64 bittisyyttä tukeva ARM arkkitehtuuri ja vasta vuoden verran markkinoilla ollut Raspberry Pi 3b+ minitietietokone, sillä uudet teknologiat vaikuttivat kiinnostavilta. Parempaan lopputulokseen ja mahdollisesti toimivaan käyttöjärjestelmään olisi voitu päästä valitsemalla arkkitehtuuriksi x86 arkkitehtuuri, joka on ollut olemassa yli kymmenen vuotta, ja jolle löytyy varmasti paljon esimerkkejä ja muuta materiaalia.

7 YHTEENVETO

Oman minimalistisen käyttöjärjestelmän tekeminen on mahdollista. Tämän todistavat useat olemassa olevat pienet käyttöjärjestelmät. Minimalistisessa käyttöjärjestelmässä on perusominaisuuksien lisäksi vain ne ominaisuudet, jotka vaaditaan käyttöjärjestelmän käyttötarkoituksessa. Ensimmäisen oman käyttöjärjestelmän tekeminen kannattaa aloittaa tunnetulla alustalla, joka on ollut markkinoilla jo useamman vuoden. Mikäli mahdollista, kannattaa ensimmäisestä omasta käyttöjärjestelmästä tehdä 32-bittinen.

Tässä työssä omaa ensimmäistä käyttöjärjestelmää lähdettiin toteuttamaan alustalle, joka oli ollut olemassa vasta muutaman vuoden, sekä valittiin arkkitehtuuri, jolle löytyy huonosti esimerkkejä. Näistä syistä johtuen työssä ei koskaan saatu valmiiksi omaa minimalistista käyttöjärjestelmää. Seuraavaa omaa käyttöjärjestelmää suunniteltaessa on otettava huomioon tässä työssä esiin tulleet hankaluudet ja tehtävä asiat toisella tavalla, jotta päästäisiin haluttuun lopputulokseen.

LÄHTEET

1. Cox, R., Kaashoek, F., Morris, M. (2014). *xv6 a simple Unix-like teaching operating system*
2. Akhmetsina, E, Gburzynski, P, Vizeacoumar, F. (2003) *PicOs: A tiny Operating system for extremely small embedded platforms*
3. Andrew S. Tanenbaum (2001). *Modern Operating Systems*
4. Rosenblum, M. (1992). *The design and implementation of a log structured file system*
5. Silberschatz, A.,Galvin, P.B.,Gagne, G. (2012) *Operating system concepts, Ninth edition*
6. Hariprasad, N.J. and Ahammed, J. (2017) *Introduction to ARM processor* [Verkkojulkaisu]. Saatavilla: https://electronicsforu.com/electronics-projects/electronics-design-guides/introduction-arm-processor [Viitattu 13.2.2019].
7. ARM (n.d) *A64 Instruction set* [Verkkojulkaisu]. Saatavilla: https://developer.arm.com/products/architecture/instruction-sets/a64 [Viitattu 20.2.2019]
8. Tung, L (2018) *Raspberry Pi 3 Model B+ arrives: Faster CPU, Wi-Fi, 300Mbps Ethernet* [Verkkojulkaisu]. Saatavilla: https://www.zdnet.com/article/raspberry-pi-3-model-b-arrives-faster-cpu-wi-fi-300mbps-ethernet/ [Viitattu 20.2.2019]
9. ARM (n.d) *Arm Processors for the Widest Range of Devices—from Sensors to Servers* [Verkkojulkaisu]. Saatavilla: https://www.arm.com/products/silicon-ip-cpu [Viitattu 20.2.2019]
10. Raspberry Pi foundation (n.d) *Raspberry Pi 3 Model B+* [Verkkojulkaisu]. Saatavilla https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf [Viitattu 20.2.2019]
11. ARM (n.d) *ARMv8 Architecture overwiev* [Verkkojulkaisu]. Saatavilla http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0677g/ada1426091785738.html [Viitattu 4.12.2019]