Dodavanje funkcionalnosti u jezgru operacijskoga sustava TempleOS

Ćorić, Viktor-Alojzije

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Viktor-Alojzije Ćorić

DODAVANJE FUNKCIONALNOSTI U JEZGRU OPERACIJSKOGA SUSTAVA TEMPLEOS

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Viktor-Alojzije Ćorić

Matični broj: 0016140113

Studij: Informacijski sustavi

DODAVANJE FUNKCIONALNOSTI U JEZGRU OPERACIJSKOGA SUSTAVA TEMPLEOS

ZAVRŠNI RAD

Mentor:

Luka Milić, mag. ing. comp.

Varaždin, rujan 2021.

Viktor-Alojzije Ćorić

Izjava o izvornosti

Izjavljujem da je ovaj moj završni rad izvorni rezultat mojega rada te da se u izradbi istoga nisam koristio drugim izvorima osim onima koji su u njem navedeni. Za izradbu rada su rabljene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredaba u sustavu FOI-radovi

Sažetak

TempleOS je besplatni 64-bitni operacijski sustav u javnoj domeni. Operacijski sustav je pisan iz temelja, a sve napisano je djelo jednoga čovjeka koji je patio od shizofrenije, Terrya A. Davisa. Unutar samoga operacijskoga sustava postoji puno novotarija, kao što su vlastiti kompilator, vlastiti programski jezik, vlastiti format za dokumente, vlastiti datotečni sustav, vlastite grafičke knjižnice pisane iz temelja, itd. Cijeli operacijski sustav radi u načinu rada *Ring 0*, te je stoga zanimljivo promatrati ga iz programerskoga pogleda. U radu se objašnjavaju osnove operacijskoga sustava, najzanimljivije značajke, dodavanje funkcionalnosti u korisnički prostor, dodavanje funkcionalnosti u jezgru, koje će se demonstrirati u vidu dodavanja hrvatskih znakova i tipkovnice, te kako preinačenu inačicu sustava dalje distribuirati.

Ključne riječi: TempleOS, jezgra, operacijski sustavi, tipkovnica, znakovi

Sadržaj

1. Uvod

TempleOS je besplatni 64-bitni operacijski sustav koji je u javnoj domeni. Autor operacijskoga sustava je američki programer Terry A. Davis, koji je sam izradio svaku sastavnicu operacijskoga sustava, od programskoga prevoditelja do slikovnih elemenata, kao i programski jezik HolyC, datotečni sustav RedSea, i sl.

Pripovijest iza samoga operacijskoga sustava je zanimljiv dio vrijedan spomena. Naime, Terry A. Davis je za vrijeme svoga života patio od shizofrenije, i često je imao deluzije različitih sadržaja, od zamišljenih prijatelja s kojima je razgovarao do agenata američke Središnje Obavještajne Agencije koji su ga pratili. U jednoj od takvih epizoda, navodno, ukazao mu se je i sam Bog, koji je htio da mu Davis izgradi Treći Hram. Bog mu je isto tako, navodno, dao i precizne specifikacije operacijskoga sustava: razlučivost 640x480, 16-bitne boje, jednokanalni audio sustav, itd. Vjerska tematika operacijskoga sustava mogla je se naslutiti iz već spomenutih HolyC (Holy See – Sveta Stolica), RedSea (Crveno more), iz samoga naziva TempleOS (Temple – Hram), ali i iz sastavnica koje će se poslije spomenuti (Adam npr.).

Operacijski sustav sveukupno sadržava 119667 redaka koda, koji uključuju jezgru operacijskoga sustava, 64-bitni programski prevoditelj, grafičke knjižnice i sva potrebna oruđa za rad sa sustavom [1]. Jedan od ciljeva operacijskoga sustava je i da smanji broj redaka koje sami programer-korisnik mora rabiti, tako da bi se napisao program *hello world* u HolyC-u, potrebno je samo napisati `"Hello World";`, a složenost se može povećavati ovisno o potrebama korisnika, mogu se dodavati funkcije kao u programskom jeziku C (gdje je za razliku od HolyC-a funkcija *main* obvezna), te se može pisati i u zbirnom jeziku.

TempleOS sve operacije obavlja u načinu rada *Ring 0*, što znači da korisnik ima pristup svemu i da nema zaštite memorije. Bilo kako bilo, jedno ponovno pokretanje računala rješava većinu problema vezanih za memoriju, kao što se je i autor sam uvjerio prigodom izradbe završnoga rada.

O potankostima operacijskoga sustava više u nadolazećim poglavljima, kao i o projektnom zadatku, koji je bio izradba tipkovnice na hrvatskom jeziku.

2. Operacijski sustav TempleOS

U ovom poglavlju će se objasniti proces instalacije operacijskoga sustava, objasniti kako se operacijski sustav rabi, te opisati najzanimljivije značajke operacijskoga sustava.

2.1. Instalacija TempleOS-a

ISO-slika operacijskoga sustava TempleOS se može preuzeti na poveznici `https://templeos.org`, gdje se preuzima `TempleOS.ISO` ili `TempleOSLite.ISO`. Nema nekoga posebnoga razloga zašto rabiti inačicu Lite, jer je puna inačica 17 MB, tako da s inačicom Lite se štedi 15 MB memorijskoga prostora, što je zanemarivo. Isto tako postoje i dopunske ISO-slike, koje sadržavaju neke dodatne stvari, kao npr. videoigru AfterEgypt, koja se ne nalazi u datoteci `TempleOS.ISO`. Dopunske datoteke se montiraju unutar sustava.

TempleOS radi na virtualnim strojevima VMWare, QEMU i VirtualBox. Autor operacijskoga sustava je rabio virtualni stroj VMWare, dok je prigodom izradbe ovoga rada rabljen virtualni stroj QEMU s modulom KVM.

Instalacija je prilično jednostavna, treba se pritisnuti `y` za instalaciju na tvrdi disk, za odabir se pritišće `y` ako se rabi virtualni stroj, pritišće se bilo koja tipka, čeka se približno jednu minutu da se instalacija dovrši, te se pritišće `y` za ponovno pokretanje. Nakon toga je cijeli operacijski sustav instaliran. Prigodom prvoga pokretanja treba pričekati par sekundâ da se dekomprimiraju pojedine datoteke, i nakon toga je operacijski sustav spreman za rad. Opcionalno se nudi vodič za porabu sustava.

2.2. Poraba TempleOS-a

Dio koji najviše zbunjuje korisnika prigodom prvoga pokretanja TempleOS-a je korisničko sučelje. Navigirati je moguće pomoću miša, ali i samo pomoću tipkovnice, pomoću strjelica, tipke `SPACE` (lijevi pritisak tipke na mišu) i tipke `ENTER` (desni pritisak tipke na mišu). U zadanom sučelju se nalaze dva otvorena prozora, a preko desnoga prozora se može vidjeti i mali sivi prozorčić koji služi kao *autocomplete*, i on korisniku izbacuje moguće prijedloge za automatsko nadopunjanje, a korisnik nadopunja započetu riječ pomoću tipke `CTRL` u kombinaciji s tipkom koja stoji uz predloženu riječ.

Da bi se za početak korisnik lakše snašao, preporuka je zatvoriti jedan prozor pritiskom na `X` u gornjem desnom kutu, ili pritiskom tipaka `SHIFT-ESC`, te drugi prozor proširiti povlačenjem njegova ruba mišem.

Pritiskom na tipke `CTRL-M` otvara se interaktivni izbornik u kojem se korisnik može dalje upoznati s operacijskim sustavom, ali i u kojem može pokretati videoigre koje je autor Terry Davis sam izradio. Iz izbornika se izlazi na tipku `ESC`.

Kroz kazala korisnik se navigira ili grafičkim putem (miš ili tipkovnica), ili kroz naredbeni redak preko naredbe `Cd;`. Sve što se upiše u naredbeni redak izravno ide u kompilator i tu se

obrađuje, to je tzv. kompilacija JIT.

Budući da privlačnost TempleOS-a ne leži u onom što operacijski sustav već nudi, već u onom što se može s njim raditi, TempleOS je ipak privlačniji programerima i hobistima nego osobama koje se bave uredskim poslovima. Stoga, to bi bilo to što se tiče „korisničkog" dijela, a u sljedećem poglavlju će se opisati naprjednija poraba kroz programiranje.

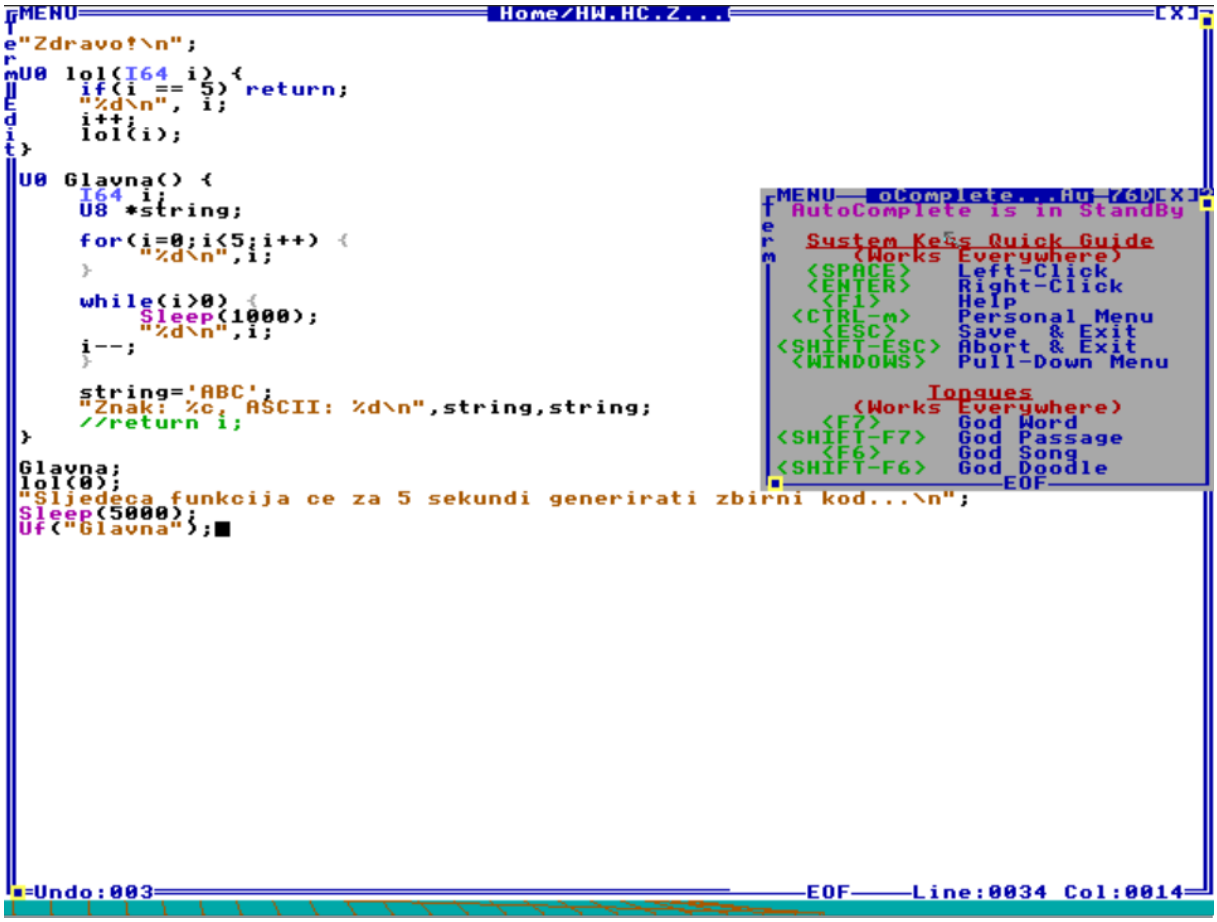
2.3. Programiranje u TempleOS-u / HolyC-u

HolyC je programski jezik koji se rabi u TempleOS-u, a kompilatori za HolyC na ostalim operacijskim sustavima (koji nisu derivati TempleOS-a) ne postoje. Jezik je osmišljen da bude što jednostavniji, slično kao i programski jezik BASIC.

Programski jezik HolyC poznaje sljedeće vrste podataka: `U0`, `U8`, `U16`, `U32` i `U64` za nepredznačene cijele brojeve veličina 0, 1, 2, 4 i 8 bajtova, `I0`, `I8`, `I16`, `I32` i `I64` za predznačene cijele brojeve veličina 0, 1, 2, 4 i 8 bajtova. Također postoji i `F64`, koja stoji za tip podataka *float* veličine 8 bajtova. Tipovi podataka za 32-bitni *float* ne postoje.[2]

Programski jezik podupire sve petlje kao i u programskom jeziku C (*for*, *while* i *do-while*), dok je uvjetna naredba *if* nešto naprjednija u usporedbi sa svojim C i C++ protulikovima (izraz `5 < i <= j + 1 < 20` je u potpunosti validan izraz, dok bi se kod C/C++-a trebalo tipkati `5 < i && i <= j + 1 && j + 1 < 20`) [2].

Na konkretnom primjeru iz Sl. 1 će se objasniti osnove programskoga jezika HolyC, a zatim će se spomenuti i naprjednije potankosti.



Slika 1. Primjer programa u jeziku HolyC

- Prvi redak jednostavno ispisuje `Zdravo!` na zaslon.
- Deklarira se funkcija `lol`, koja je zamišljena kao rekurzivna funkcija. Odbrojava do 5 sve dok ne dođe do sidrenoga uvjeta (engl. *anchor case*).
- U funkciji `Glavna` se pokazuje kako se deklariraju cijeli 64-bitni brojevi, te kako se deklarira pokazivač. Nakon deklaracije se pokazuje rad s petljama, petlja *for* odbrojava od `0` do `4`, dok petlja *while* odbrojava od `5` do `1` (tu se još demonstrira i funkcija `Sleep`). Nakon toga se inicijalizira znakovni niz, i ispisuje se njegova stvarna i brojčana vrijednost. Funkcija može i vraćati određene podatke.
- Nakon deklaracije funkcije `Glavna` dolazi pozivanje tih funkcija. Kao što se vidi, funkcija `lol` prima jedan parametar. Funkcija `Glavna` se može pozivati sa zagradama i bez njih.
- Funkcija `Uf` obilježava `Unassemble function`, te vraća kod rečene funkcije u zbirnom jeziku.

HolyC isto tako podupire i koncepte unija i razreda. Unija djeluje na isti način kako i djeluje u C/C++-u, dok je koncept razreda ostvaren kao nešto između C/C++-ovoga `structa` i C++-ova razreda, jer postoji i svojstvo nasljeđivanja povrh običnoga `structa`.

Postoje dvije vrste kompilacije u TempleOS-u, a to su Just In Time (JIT) i Ahead Of Time (AOT). Kompilacija JIT se rabi za veliku većinu svih radnjâ u operacijskom sustavu, jedini dio koda koji se prevodi Ahead of Time (AOT) je jezgra operacijskoga sustava, za koju se prigodom prevođenja stvara binarna datoteka. Kompilacija JIT omogućuje da se HolyC rabi i kao naredbeni redak tj. ljuska operacijskoga sustava. Svaka naredba koja se upiše izravno ide u programski prevoditelj. Razlog toga je što autor operacijskoga sustava nije bio ljubitelj razlikovanja između sintakse programskih jezika (C, C++) i naredaba za upravljanje operacijskim sustavom (`bash`).

Neke zanimljivosti i novosti u programskom jeziku HolyC su:

- Funkcije bez argumenata ne trebaju zagrade.
- Argumenti sa zadanim vrijednostima ne moraju se deklarirati na kraju.
- Uvjeti se mogu staviti pod jedan izraz, npr. `5 <= x < y < 15 == z`, umjesto 4 izraza koja bi kombinirali s operatorom `&&` u C-u i C++-u [2].
- *Switch* može imati i pod-*switch*, *switch* unutar *switcha*, rabeći ključne riječi *start* i *end* [2].
- `U0` tip podatka, koji je jednakovrijednica tipu podatka `void` u C-u i C++-u zauzima 0 bajtova, za razliku od `voida` u GNU-ovu dijalektu C-a koji zapravo zauzima jedan bajt.
- Funkcije imaju ugrađene varijable za brojenje argumenata i čitanje istih (`argc` i `argv`), te djeluju na sličan način kao što djeluje operator *this* u jeziku C++ [2].
- HolyC ima potporu za *try-catch-throw*.
- Direktiva `#exe`, koja omogućuje izvršavanje naredbe i pohranjivanje njezine vraćene vrijednosti unutar programskoga koda.
- Sve vrijednosti se proširuju u 64 bita kada im se pristupa [2].

2.4. DolDoc

DolDoc je TempleOS-ova vrsta dokumenta. Ono što je vrlo zanimljivo je da su svi izbornici koje korisnik vidi pisani u obličju DolDoc, a u to se može i uvjeriti pritiskom `CTRL+T` tipaka, koje će prebacivati između čistoga teksta i teksta s grafikama.

Ono što je specifično za obličje DolDoc je što se pored izvornoga koda programa TempleOS u nj mogu pohranjivati i 3D grafike, slike, itd., i program će se i dalje uspješno prevoditi i pokretati. Usporednica tomu bi bila kada bi se LibreOffice ili Microsoft Word rabio kao razvojna okolina.

Sve naredbe u DolDocu se moraju opkoliti znakom dolara, dakle primjer takve naredbe bi bio `$CL$`, naredba za brisanje zaslona koja je dosta slična funkciji `DocClear`, čija bi jednakovrijednica za Microsoft Windows bila `cls`, a jednakovrijednica za UNIX bila `clear`.

Naredaba ima previše, pa da se ne navode sve u radu, ako čitatelja zanima može ih pregledati u samom TempleOS-u pod `/Doc/DolDocOverview.DD.Z` (preporučljivo zbog organiziranosti), ili na referenciji [3].

Slika 2. Primjer dokumenta DolDoc, na lijevoj slici je prikazan normalni prikaz, na desnoj slici je prikazan tekstualni prikaz

2.5. RedSea

RedSea je datotečni sustav koji je Terry A. Davis napisao. TempleOS pored RedSea podupire i datotečne sustave FAT32 i ISO9660.

RedSea je jednostavni 64-bitni datotečni sustav koji je sličan datotečnom sustavu FAT32, ali umjesto grozdova rabi apsolutne blokove. FAT (*File Allocation Table*) tablica nije implementirana jer je patentirana, te umjesto nje se rabi alokacijska bitmapa [4].

Glavna prednost datotečnoga sustava RedSea je jednostavnost. Autor TempleOS-a često navodi Commodore 64 kao svoju nadahnutost za većinu toga u operacijskom sustavu, te datotečni sustav RedSea tu nije iznimka. Autor se je dojmio kako je korisnik Commodorea 64 mogao izravno pristupiti diskovnim blokovima i kako je mogao povratiti obrisane datoteke iz tih blokova.

RedSea rabi alokacijsku bitmapu susjednih datoteka, ne prati pokvarene blokove i zalihosne FAT-ove. Susjedna alokacija je iznimno jednostavna i brza, ali nije toliko efikasna jer

je takav datotečni sustav skloniji fragmentaciji i dosta se prostora troši nepotrebno [5]. Kod TempleOS-a to nije problem doduše jer je sam autor zabranio da se rabe multimedijske datoteke [6] [7], navodeći fragmentaciju memorije kao ozbiljan problem.

Još jedna zanimljiva značajka datotečnoga sustava RedSea je što on sam komprimira sve datoteke koje završavaju proširkom `.Z`. [6] Kada se pokuša otvoriti datoteka koja završava s proširkom `.Z`, uređivač teksta *vim* javlja pogrješku da se radi o komprimiranoj datoteci, iako je dio teksta donekle čitljiv.

Sada kada su spomenute i opisane najzanimljivije značajke TempleOS-a, može se početi s opisivanjem praktičnoga dijela ovoga rada.

3. Dodavanje funkcionalnosti u TempleOS

Funkcionalnosti se u operacijski sustav TempleOS mogu dodavati na dva načina, unutar korisničkog prostora, koje se može obavljati na dva načina, i unutar jezgre.

Treba voditi računa da se funkcionalnosti u korisničkom prostoru prevode Just In Time (JIT), dok se funkcionalnosti u jezgri prevode Ahead Of Time (AOT).

3.1. Dodavanje funkcionalnosti u korisnički prostor

Dodavanje funkcionalnosti u korisnički prostor unutar TempleOS-a je jednostavno kao i na bilo kojem drugom operacijskom sustavu, jedina razlika dodavanja funkcionalnosti u TempleOS-u je to što od razvojnih oruđa ima dostupne samo zbirni jezik i programski jezik HolyC za programiranje, dok u drugim operacijskim sustavima većinom ima širi spektar razvojnih alata.

Za dodavanje funkcionalnosti u korisnički prostor je potrebno prvo napisati željeni program. Primjer koji će biti prikazan u svrhu prikaza dodavanja programa je primjer sa Sl. 1.

Postoji više načina kako dodati funkcionalnost u korisnički prostor operacijskog sustava TempleOS, a to su:

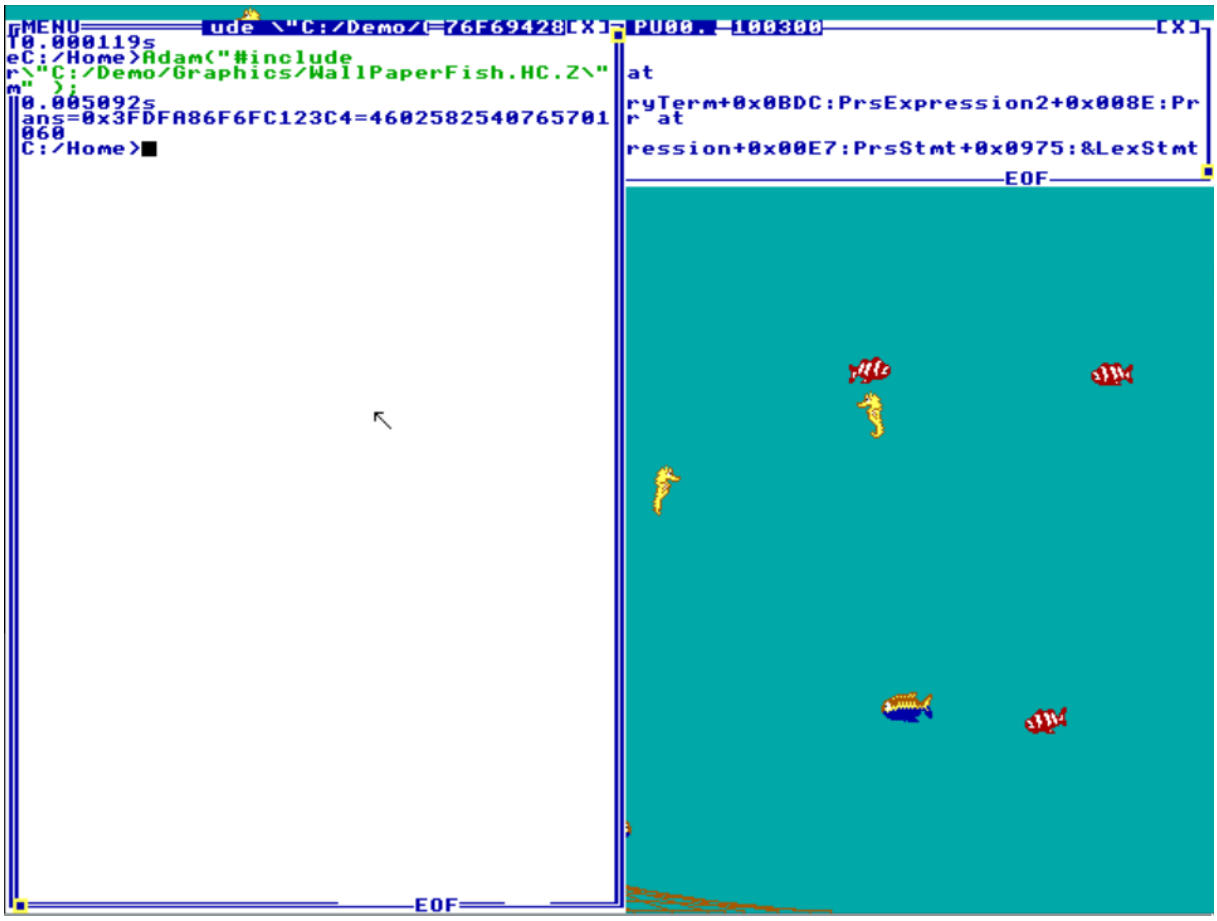
- Otvaranje datoteke u uređivaču teksta i pritisak tipke `F5`.
- Pokretanje pomoću direktive `#include`.
- Pokretanje programa tijekom prevođenja nekog drugog programa pomoću direktive `#exe`.
- Pokretanje datoteke unutar Adama (trajni kod).

Pokretanje pomoću tipke `F5` unutar uređivača teksta je jasna sama po sebi, potrebno je otvoriti željenu datoteku i pritisnuti `F5`.

Pokretanje pomoću direktive `#include` se vrši tako što unutar ljuske operacijskoga sustava se piše naredba `#include "<naziv datoteke bez proširke>";`. Isto tako, budući da sve naredbe iz ljuske idu izravno u kompilator i prevode se Just In Time (JIT), ovakav način pokretanja programa se može rabiti i unutar drugoga programa.

Operacijski sustav TempleOS nudi i jednu zanimljivu značajku, a to je direktiva `#exe`, koja omogućuje izvršavanje koda tijekom procesa prevođenja, te se taj kod može umetnuti u tok koji se prevodi [8]. To je HolyC-ova alternativa za makronaredbe znane iz programskih jezika C/C++.

Dodavanje funkcionalnosti u korisnički prostor se može odraditi na još jedan način, koji je više križanac između korisničkog i sustavskog prostora, a to je pomoću Adama, koji je prvi proces koji se stvara i koji se ne može ugasiti.



Slika 3. Program uključen preko Adama, postavlja dinamičku pozadinu koja se miče

3.2. Dodavanje funkcionalnosti u jezgru

Dodavanje funkcionalnosti u jezgru je nešto zahtjevniji zadatak, ali opet je puno jednostavnije u usporedbi s drugim operacijskim sustavima.

Prigodom dodavanja funkcionalnosti u jezgru operacijskoga sustava TempleOS treba voditi računa da je jezgra jedini dio TempleOS-a koji se prevodi Ahead Of Time (AOT), te da se sve operacije unutar jezgre uvode u načinu rada kojeg autor operacijskoga sustava naziva *raw mode*. Programiranje je drugačije nego u korisničkom prostoru jer se nema fleksibilnost koju nudi način prevođenja Just In Time. Primjerice, kada bi se deklarirala funkcija koja se izvršava u jezgri koja bi na pritisak tipke izvršila neki program unutar korisničkog prostora, prigodom prevođenja jezgre bi se dogodila pogrješka, jer se ne bi moglo pozicionirati na odgovarajuću lokaciju da bi se izvršila rečena datoteka, jer jezgra nema predznanje o ostalim dijelovima koda koji se ne prevode načinom AOT.

Kao što je već spomenuto u odlomku iznad, sav napisani kod u jezgrenom prostoru potrebno je prevesti načinom AOT. To se postiže pomoću funkcije `BootHDIns;`, koja će se potanje objasniti u nastavku.

Slika 4. Prevođenje jezgre – osnovne postavke

Kao što je vidljivo na Sl. 4 i 5, za prevođenje jezgre su za neke stvari dovoljne postavke, ali ipak je potrebna i određena količina ljudskog međudjelovanja za uspješno prevođenje. Ono što korisnik treba u velikoj većini slučajeva učiniti jest odabrati disk čija se jezgra prevodi, a to su većinom diskovi `C` i `D`, ovisno o tom što je odabrano prigodom dizanja operacijskoga sustava. Zatim je potrebno pritisnuti tipku `p` da bi saznale informacije o trenutnom disku, te iz tih informacija potrebno je ili pritisnuti broj koji se nalazi lijevo od diska, ili ručno upisati podatke o disku, odnosno njegov *Base0*, *Base1*, te *Unit*. U primjeru na Sl. 4 odlučeno je ručno upisati podatke o disku. U primjeru na Sl. 5 su prikazane naprjednije opcije dostupne prigodom prevođenja jezgre, ali su one ostavljene na zadanim vrijednostima, a slika je uključena samo da bi se prikazalo koje su to naprjednije postavke.

Slika 5. Prevođenje jezgre – naprjednije postavke

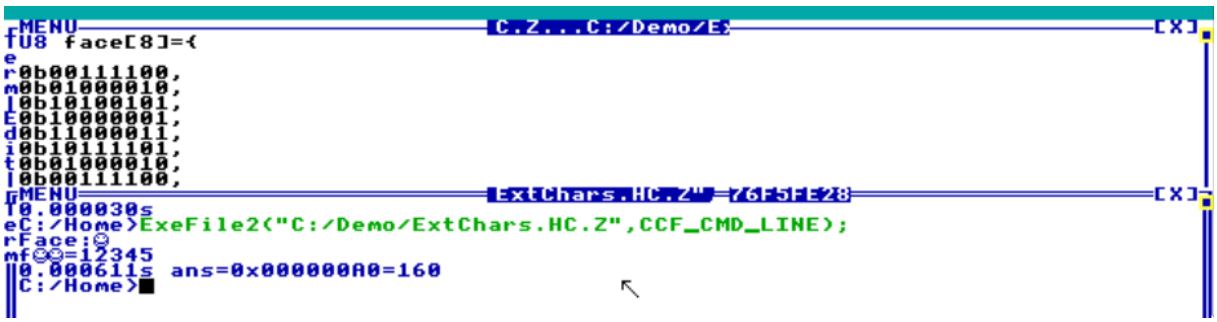
4. Dodavanje hrvatske tipkovnice u TempleOS

Praktični zadatak ovoga rada je bio ugraditi hrvatske znakove i tipke u jezgru TempleOS-a, ali je isto tako i dio praktičnoga dijela izraditi usporedbu dodavanja tipkovnice u jezgru s dodavanjem tipkovnice unutar korisničkoga prostora. U sljedećim odjeljcima će se opisati oba procesa, počevši s dodavanjem znakova i tipaka unutar korisničkoga prostora.

4.1. Dodavanje u korisnički prostor

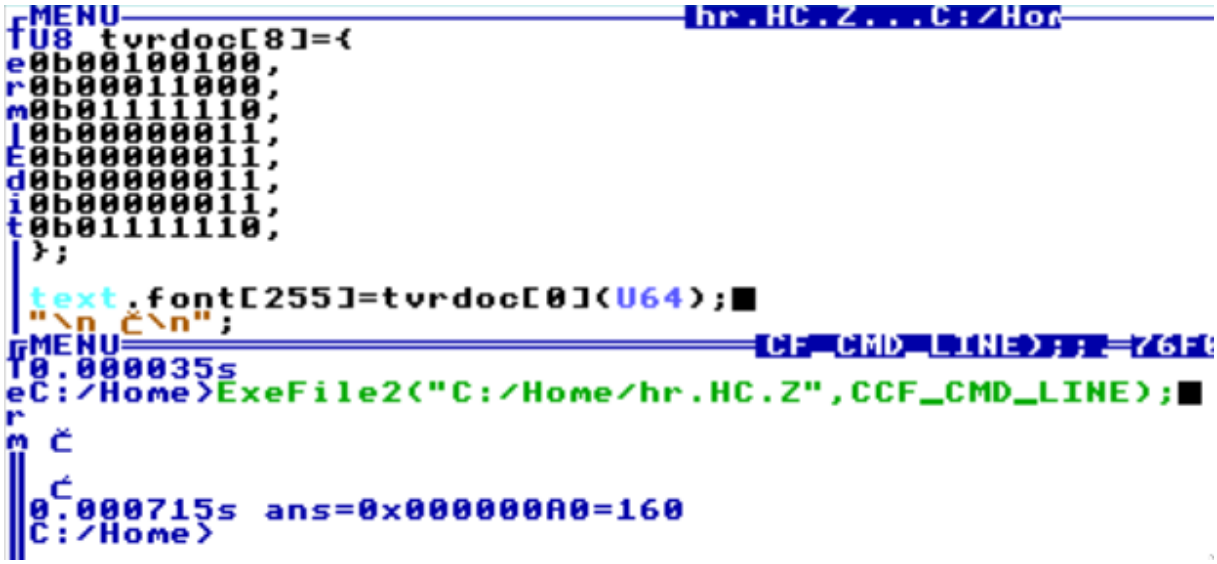
4.1.1. Stvaranje vanjskih znakova

Stvaranje vanjskih znakova u TempleOS-u je dosta jednostavno. Uz mali mig iskusnijih programera TempleOS autor je doznao da u kazalu `/Demo` postoji datoteka koja se zove `ExtChars.HC.Z`. U toj datoteci je dokumentiran proces izradbe vanjskoga znaka. U primjeru koji je izradio g. Davis se je nalazilo polje `face` od osam 8-bitnih nepredznačenih cijelih brojeva. Može se opaziti da jedinice u takvu zapisu predstavljaju jedan *pixel*, da zajedno čine smješka, a da su znakovi zapravo bitmape dimenzije 8x8 bitova.



Slika 6. Zadani primjer unutar datoteke `ExtChars.HC.Z`

Ovaj primjer je malo nezgodan jer se na njem ne može opaziti jedan problem koji će se prikazati na Sl. 7, a to je da je zapis *little-endian*, pa se trebaju izvrnuti bitovi u redcima. U primjeru sa smješkom to se nije vidjelo jer je smješko simetričan po y-osi, pa je u svrhu prikaza izrađen preinačeni primjer koji prikazuje istu stvar, ali na kojem se može opaziti izvrnutost x-osi, a primjer prikazuje hrvatski znak `Č`.



Slika 7. Preinačeni primjer vanjskoga znaka

4.1.2. Promjena rasporeda tipaka

Raspored tipaka unutar korisničkog prostora najjednostavnije se može odraditi tako da se izrade svi znakovi postupkom opisanim u odjeljku 4.1.1, da se pohrane u jednu datoteku i da se izvrši `#include` te datoteke. Kada se izrade svi potrebni znakovi, potrebno je promijeniti raspored tipaka, da ne bi došlo do zbrke u semantici prigodom izmjenjivanja jezika. Raspored se može odrediti definiranjem dekodne tablice u zbirnom jeziku, koja se ostvaruje jednostavno kao polje 8-bitnih znakova. Preinačene znakove je moguće ubaciti pomoću kombinacije tipaka `CTRL-ALT-A`, pozicioniranja na željeni znak, i pritiska `SPACE`-tipke. Nakon izrade dekodnih tablica, rečene je potrebno ubaciti u memoriju, što se radi pomoću funkcije *MemCpy*.

Slika 8. Primjer dekodne tablice i njezina učitavanja u memoriju

4.2. Dodavanje u jezgru

4.2.1. Stvaranje *fonta* unutar jezgre

Unutar kazala `/Kernel` mogu se opaziti datoteke `FontStd.HC.Z` i `FontCyrillic.HC.Z`. Uz pomoć izvrsne funkcije `Find` dostupne u TempleOS-u, lako se može otkriti da postoji mogućnost da se u *fontu* dodaju ćirilični znakovi na pritisak tipaka `CTRL-ALT-F` (QEMU će to zabilježiti kao naredbu za cijeli zaslon i ne će utjecati na TempleOS, pa je stoga potrebno učiniti sljedeće: prisitsnuti `CTRL-ALT-2`, upisati `sendkey ctrl-alt-f`, pritisnuti `CTRL-ALT-1`). Ali prvo se treba opisati proces stvaranja *font*-datoteke , pa se onda baviti dodavanjem novih kombinacija tipaka.

Prvo treba kopirati datoteku `FontStd.HC.Z` i nazvati ju po želji. Autor ju je nazvao `FontHr.HC.Z`. Na prvi pogled izgleda kao da se radi o apsolutnim memorijskim adresama gdje su znakovi definirani, iako adrese izgledaju dosta slučajne. Nakon neuspješne potrage gdje su te adrese preslikane, gdje u kodu se definira *font*, i sl., autor se je dosjetio da to uopće nisu adrese, nego da je to heksadekadski prikaz tih znakova, gdje dva znaka predstavljaju jedan bajt.

Ono što se razlikuje od dodavanja u korisničkom prostoru je što su u jezgri osi x i y suprotne od intuitivnoga očekivanja korisnika, dok je kod korisničkoga dodavanja vanjskoga znaka samo x-os izokrenuta (*little endian*). To se najzornije može opaziti na Sl. 9, gdje se može vidjeti kako je slovo `Č` zakrenuto za 180°.

Slika 9. Primjer znaka u heksadekadskom zapisu

4.2.2. Obavljanje radnjâ na pritisak tipke

Sljedeći očigledan korak koji se mora ostvariti je obavljanje radnjâ na pritisak tipke, slično kao što se mora pritisnuti kombinacija tipaka da bi se prebacilo na ćirilični *font*. Prvi korak koji se čini je pogled u dokumentaciju, u kazalo `/Doc` u TempleOS-u. Tu se mogu opaziti dvije datoteke koje počinju s `Key*`. Iz datoteke `KeyAlloc.DD.Z` se može opaziti da su kombinacije tipaka `ALT` i `ALT-SHIFT` slobodne da ih korisnik dodijeli kako želi, ali da se kombinacija `CTRL-ALT` obrađuje na sustavskoj razini pomoću funkcije `CtrlAltCBSet()`. Autor TempleOS-a je uzeo za pravo da predbilježi tipke kako on želi, što je stvaralo sukobljenost u izradbi završnog rada, te je donešena odluka autora da će se rabiti tipka `ALT` umjesto `CTRL-ALT`. Tipka `ALTGR` ima isti *scan code* kao i tipka `ALT` unutar operacijskoga sustava.

Pomoću funkcije `Find` se može potražiti funkcija `CtrlAltCBSet`, gdje se ona rabi, i nakon pretrage da se opaziti da se dosta rabi u datoteci `/Kernel/KeyDev.HC.Z`.

Funkcije su izvrsno imenovane, stoga se lako da pronaći funkcija koja je odgovorna za izmjenu standardnoga i ćiriličnoga *fonta*, i tu se mogu pronaći korisni tragovi kako dalje s

projektnim zadatkom.

Za ispitivanje je dodana kombinacija tipaka `CTRL-ALT-i` koja bi samo ispisala tekst `Hello World` na zaslon. Nakon prevođenja jezgre preko funkcije `BootHDIns;` sve je radilo kao i očekivano. Nakon toga treba kopirati i izmijeniti dio koda odgovoran za izmjenu *fontova* (to je redak `SwapI64(&text.font, &text.font_hr);`) i alocirati ga za `CTRL-ALT-p`, te je taj dio za sad gotov, ali poslije će se vratiti na nj u odjeljku 4.2.5.

4.2.3. Dodavanje *fonta* u memoriju

Iako je datoteka `FontHr.HC.Z` u jezgri stvorena, nekako ju je trebalo učitati u samu jezgru da bi se prepoznala u procesu kompilacije AOT jezgre. Važne lokacije za inicijaliziranje *fonta* su vidljive na Sl. 10.

Koristeći se funkcijom `Find` mogu se pronaći dva važna mjesta gdje se rabi pokazivač iz funkcije `SwapI64` iz prethodnoga poglavlja, a to su `KMain.HC.Z` i `KernelA.HH.Z`.

U `KernelA.HH.Z` treba dodati odgovarajući pokazivač na hrvatski *font*, koji čini globalnu varijablu za tekst, a u datoteci `KMain.HC.Z` toj varijabli treba dodijeliti naziv polja iz naše *font*-datoteke.

Sada se na pritisak tipaka `CTRL-ALT-p`, kao i iz naredbene ljuske, mogu mijenjati *fontovi*. Budući da je sav tekst u TempleOS-u na engleskom jeziku, vidne promjene nisu opazive na prvi pogled, ali se mogu opaziti tek otvaranjem znakovnoga izbornika na pritisak `CTRL-ALT-a`.

```
C:/Kernel>F("aux_font");
C:/Demo/Graphics/FontEd.HC.Z,0008                text.aux_font=my_font;
C:/Demo/Graphics/FontEd.HC.Z,0010 <CTRL-ALT-f> will toggle main font and
aux_font.
C:/Kernel/KMain.HC.Z,0068    text.aux_font=sys_font_cyrillic;
C:/Kernel/KeyDev.HC.Z,0152    SwapI64(&text.font,&text.aux_font);
C:/Kernel/KernelA.HH.Z,3546    U64        *font,*aux_font,*hr_font;
0.565148s ans=0x00000005=5
C:/Kernel>F("sys_font_cyrillic");
C:/Kernel/FontCyrillic.HC.Z,0003 U64 sys_font_cyrillic[256]= {
C:/Kernel/KMain.HC.Z,0068    text.aux_font=sys_font_cyrillic;
0.563275s ans=0x00000002=2
C:/Kernel>
```

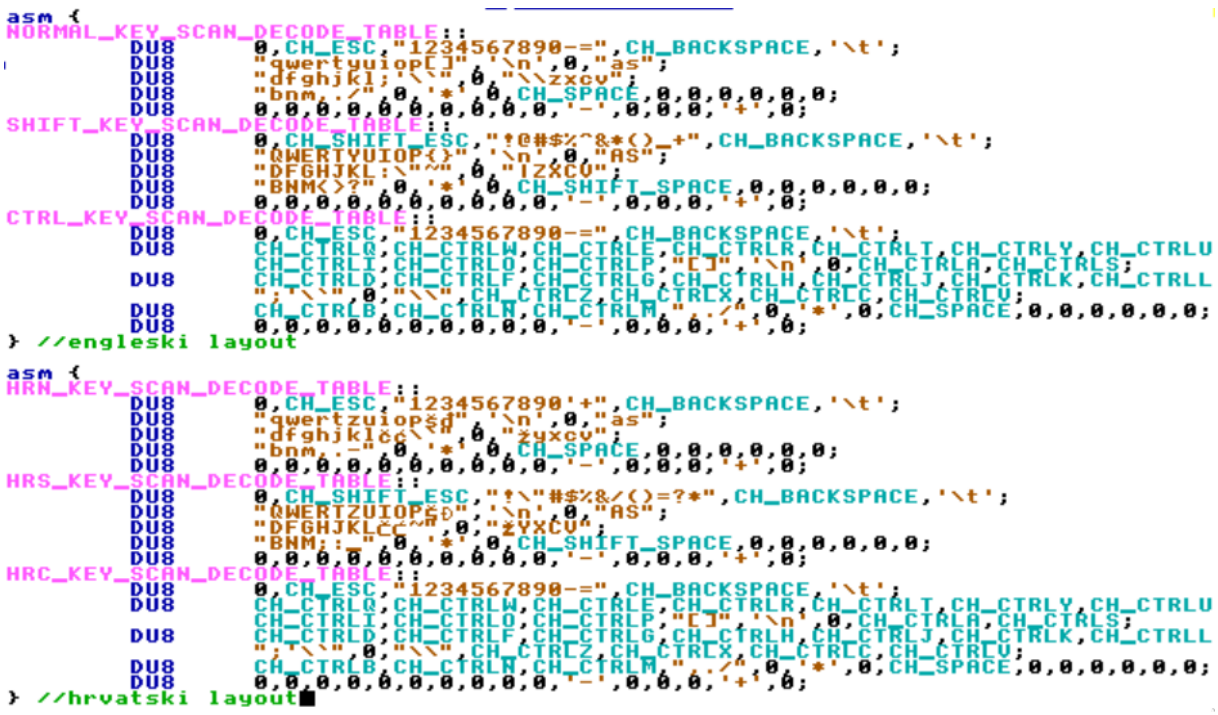
Slika 10. Sve važne lokacije gdje se *font* inicijalizira u memoriji (primjer s ćiriličnim *fontom*)

4.2.4. Dodavanje novoga rasporeda tipaka

Izmjena *fonta* nije sama po sebi dovoljna prigodom izradbe tipkovnice. Na primjer, kada bi se imao tekst na engleskom jeziku koji bi glasio `yes;`, i kada bi se samo izmijenio *font* dobio bi se ozbiljan problem, na hrvatskom bi taj tekst u potpunosti izgubio značenje, glasio bio `zesč`. To je zbog temeljno drukčijega rasporeda tipaka (eng. *layout*) između tih dviju tipkovnica. U engleskoj tipkovnici `Z` se nalazi na položaju `Y`, znak `;` se nalazi na položaju slova `Č`, i sl. Stoga je bilo potrebno locirati datoteku gdje je raspored tipaka definiran, i uz malo kopanja po kazalu `/Kernel` može se vidjeti da se raspored tipaka nalazi u datoteci `/Kernel/SerialDev/Keyboard.HC.Z`.

Ispitivanja radi, mogu se izmijeniti položaji slova `Z` i `Y`, ponovno prevesti jezgra, i ispitati radi li sve kako treba. Nakon uspješnoga ispitivanja valja vratiti datoteku u izvorno stanje, te ju kopirati u datoteku s nazivom po izboru. Autor je rabio naziv `KeyHr.HC.Z`. U toj datoteci se

dodaju hrvatski znakovi na mjesto gdje pripadaju. Također se treba izraditi i datoteka koja bi služila čisto za izmjenu varijabli. Autor ju je nazvao `Temp.HC.Z`. Ta datoteka postoji iz razloga kad bi se hrvatsla tipkovnica kopirala u englesku, raspored tipaka engleske tipkovnice se ne bi mogao dobiti natrag. Učinjene izmjene su vidljive na Sl. 11.



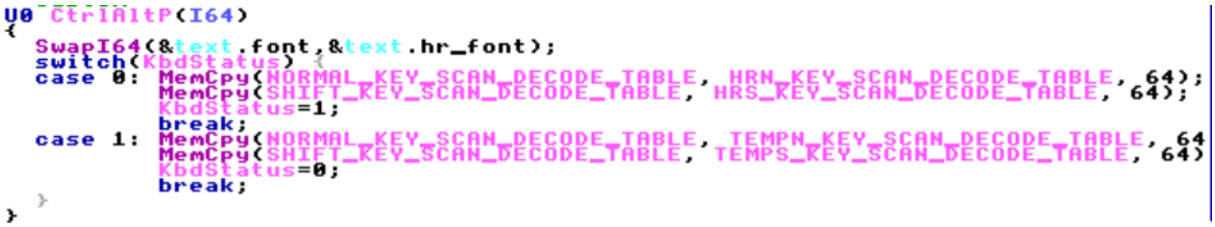
Slika 11. Usporedba rasporeda tipaka

Prevođenje jezgre u trenutačnoj fazi bi davalo pogrješke, jer su funkcije definirane u `Keyboard.HC.Z` već definirane. Kada bi se pokušale obrisati iz kopirane datoteke, to bi isto tako imalo problema što se tiče dinamičkoga izmjenjivanja tipkovnice. Javio se je problem gdje su vrijednosti iz zbirnoga jezika u navedenim datotekama morale biti prevedene prije datoteke `KeyDev.HC.Z`, ali jednostavni `#include` prije `KeyDev.HC.Z`-a ne bi radio jer bi funkcije koje se nalaze u datoteci `Keyboard.HC.Z`-a zahtijevale funkcije iz datoteke `KeyDev.HC.Z`. Ukratko i pojednostavljeno, A jepotrebno uključiti prije B, a B je potrebno uključiti prije A, što je proturječno. O tome više u poglavlju 4.2.5.

4.2.5. Dinamičko izmjenjivanje tipkovnice

Kada su svi koncepti iznad izrađeni, preostaje još dodati i dinamičko izmjenjivanje tipkovnice, odnosno da se na pritisak kombinacije tipaka izmijene *font* i raspored tipaka.

Proces dodavanja kombinacije `CTRL-ALT` je opisan u odjeljku 4.2.2, ali u ovom odjeljku će se potanje obrazložiti što se sve događa unutar kombinacije `CTRL-ALT-p`, za koju je odlučeno da će biti okidač za dinamičku izmjenu tipkovnica. Dio koda koji se pokrene pri pritisku kombinacije tipaka `CTRL-ALT-p` će se potanje razmotriti u Sl. 12.



Slika 12. Funkcija za dinamičko izmjenjivanje tipkovnice

Funkcija `SwapI64`, odnosno kako se ista rabi za izmjenu *fonta*, je već opisana u poglavlju 4.2.2. Ona se obavlja pri svakom pozivu, tako da nije potrebno stavljati ju u `switch`. `KbdStatus` je globalna varijabla koja je jednostavno inicijalizirana na ništicu na samom početku datoteke `KeyDev.HC.Z`, a ona služi za praćenje stanja, da vidi koja je tipkovnica trenutačno djelatna.

Potom se odvija kopiranje jednoga dijela memorije u drugi, odnosno kopira se vrijednost koja je navedena u zbirnom kodu kod definiranja *layouta*. To se obavlja uz pomoć funkcije `MemCpy`. U programskom jeziku HolyC se ne mogu proslijediti strukture veće od 8 bajtova kao što se mogu u C-u, stoga je odlučeno da će se rabiti `MemCpy` veličine 64 bajta. Ako se pogleda Sl. 11, može se opaziti da se dio u zbirnom jeziku sastoji od 5 polja 8-bitnih nepredznačenih cijelih brojeva, znači svaki znak se sastoji od jednog bajta. Ručnim brojenjem tih znakova može se doći do zaključka da struktura zauzima između 75 i 80 bajtova, ali isto tako može se opaziti da su zadnji redci isti, te ih nije potrebno kopirati, i tim se je količina podataka za kopiranje smanjila za 16 bajtova.

`NORMAL_KEY_SCAN_DECODE_TABLE` obilježava tablicu koja je djelatna, a `HRN_KEY_SCAN_DECODE_TABLE` obilježava hrvatsku tipkovnicu (`HRN` znači `HR Normal`). Ista stvar se radi i sa tipkama koje se dobiju uz tipku `SHIFT`, u `SHIFT_KEY_SCAN_DECODE_TABLE` se kopiraju vrijednosti iz `HRS_KEY_SCAN_DECODE_TABLE`. `CTRL_KEY_SCAN_DECODE_TABLE` nije potrebno kopirati jer su to funkcijske tipke koje su iste za svaki jezik, npr. kombinacija tipaka `CTRL-B` gasi i pali rubove prozora.

Ovim procesom je izrađena velika većina tipkovnice, ali valja opaziti da još nedostaju `ALT` tipke, kako bi prikazali znakove poput `\`, `@`, €, `<`, `>`, `|`, i sl. Više o tom u odjeljku 5.1.

4.2.6. Dodavanje `ALT`-tipaka

Iako se čini kao dosta trivijalan zadatak, dodavanje `ALT`-tipaka se je pokazalo zahtjevnijim problemom nego što se čini. Naime, prvi pokušaj ostvarivanja kombinacije `ALT`-tipaka je bio ubacivanje istih u kod zbirnoga jezika koji je definirao i raspored tipaka. Pomoću funkcije `Find` da se opaziti da su konstante definirane u zbirnom jeziku automatski dostupne u svim dijelovima jezgre, kao što je i već viđeno u datoteci `KeyDev.HC.Z`. Drugo spominjanje tih konstantata se je spominjalo u datoteci s tipkovničnim funkcijama koje su već ranije u radu logički odvojene od zbirnoga koda, a to je `Keyboard1.HC.Z`. Nakon dosta pokusâ s tom datotekom, odnosno s `ScanCode2Char` funkcijom (kao i s funkcijom `Char2ScanCode`), iz nekoga razloga jednostavno nije radilo, pa je odlučeno da će se tražiti alternativno rješenje.

Alternativno rješenje je bilo uzeti datoteku `HomeKeyPlugIns.HC.Z`, koja se nalazi u korijenskom kazalu TempleOS-a, te ju staviti u Adam, te ju `#include`-ati prigodom paljenja operacijskoga sustava. Budući da je Adam prvi proces koji se stvara i ne može se ugasiti, on se može smatrati sustavskim dijelom operacijskoga sustava. Jedina razlika je što se on obavlja kompilacijom Just In Time, za razliku od jezgre, koja je jedini dio sustava koja se prevodi Ahead Of Time.

U datoteci `/Doc/KeyDev.HC.Z` je opisano kako se pomoću funkcije `KeyDevAdd` dodaju nove funkcionalnosti za tipkovnicu, a za svrhe ovog rada najbolje bi bilo rabiti korisnički rukovatelj `MyPutKey` jer je on od svih opisanih rukovatelja najprikladniji za operacije potrebne za izradbu ciljanog zadatka. Na Sl. 13 i 14 je prikazan proces dodavanja tih tipaka, dok je proces dodavanja *fonta* već opisan u poglavlju 4.2.1.

Kod vidljiv na Sl. 13 je sam po sebi jasan. Prvi uvjet *if* govori da tipka `ALT` mora biti pritisnuta, te da tipka `CTRL` ne smije biti pritisnuta. `CTRL` uvjet služi samo zato da ne bi bilo nekih konfliktnih kombinacija koje bi obavljale dvije radnje u isto vrijeme, kao npr. u slučaju kombinacije `CTRL-B`, gdje bi se napisao znak `{`, ali bi se i uklonile granice prozora TempleOS-a. Ostatak se svodi na to da ovisno o tom koji znak TempleOS primi, i ako taj *scan code* nema opis u sustavu, da stavi neki drugi znak.

Na Sl. 14 postoji jedan dio koda koji nije jasan na prvi pogled, pa ga valja objasniti. To je uvjet koji glasi `if(sc&0x7F==0x56)`. Razlog zašto se obavlja logičko I se nalazi u datoteci `/Doc/CharOverview.HC.Z`. U datoteci se može opaziti opis povratne vrijednosti pritisnute tipke, odnosno da ništični bajt sadržava *scan code*, te da bi se on dohvatio potrebno je se nekako riješiti zastavice ispred, a to se može obaviti pomoću trunkacije viših bajtova, koja se može odraditi bitovnim I. Budući da se na tipkovnici rabe ASCII znakovi od 0 do 127, može se rabiti heksadekadski broj `7F`, jer on kad se zapiše binarno sadržava 7 jedinica, tako da će se sve iznad toga truncirati. Nije pogrješka rabiti heksadekadski broj `FF`, jer ASCII u TempleOS-u je 8-bitni, ali budući da se na tipkovnici samo nalaze znakovi od 0 do 127, dok se znakovima s višim vrijednostima može pristupiti samo pomoću kombinacije tipaka `CTRL-ALT-a`.

Razlog zašto se rabi `0x56` je što je to *scan code* tipke kojom se pišu znakovi `<`, `>` i `|`. Njezin *scan code* se može doznati preko datoteke `MsgLoop.HC.Z` koja se nalazi u kazalu `/Demo`.

Dodavanjem te tipke je pokrivena cijela tipkovnica. Funkcijom `KeyDevAdd` se dodaju izrađene tipke `ALT`, te je još nužno obaviti `#include` naše datoteke u datoteci `MakeAdam.HC.Z` u kazalu `/Adam`, i tim je projektni zadatak gotov.

Slika 13. Funkcija `MyPutKey` u kojoj se definira ponašanje tipkovnice

Slika 14. Ostatak funkcije `MyPutKey` i funkcija `KeyDevAdd` gdje se dodaje funkcija `MyPutKey` kao tipkovnični uređaj

5. Prevođenje, pokretanje i ispitivanje sustava

U posljednjem poglavlju završnoga rada se obrađuje proces prevođenja i pokretanja izmjenjene inačice TempleOS sustava.

5.1. Prevođenje preinačenoga TempleOS-a

Kada je cijela tipkovnica napravljena, postavlja se pitanje kako se napravljeno može podijeliti s drugima. Jedan izbor bi svakako bio podijeliti sliku virtualnoga stroja s drugima, ali to bi zauzimalo onoliko koliko je autor sam odredio prostora, u ovom slučaju 3 gigabajta, što je masivno i nepraktično za dijeljenje.

Bolji izbor je rabiti program `DoDistro.HC.Z` koji je dostupan u samom TempleOS-u, pod kazalom `/Misc`. `#include`-anjem te datoteke se kopira sve što se nalazi u operacijskom sustavu, pa čak i rječnik, koji opcionalno može obrisati da bi se uštedjelo na prostoru (ovom primjeru je uključen i rječnik, te je ukupna veličina ISO-datoteke 31 MB). Nakon što se stvori ISO-datoteka, može se locirati u kazalu `/Tmp`.

Kada je stvorena vlastita distribucija, postavlja se pitanje kako ISO-datoteku koja se nalazi unutar TempleOS-a prenijeti izvan operacijskoga sustava, budući da TempleOS ne podupire mrežne operacije. Nakon malo pokusiranja, pa zatim istraživanja, pronađena su dva načina, ovisno o vrsti diska koja se ima, je li to obličje *raw* ili *qcow2* [9].

Ako se ima obličje slike *raw*, naredba koja se pokreće jest:

```
#!/bin/sh
# montiranje
sudo mount -o loop,offset=32256 staza/do/slike staza/montiranja
```

Ispis 1. Montiranje obličja slike *raw*

Ako se pak radi o obličju slike *qcow2*, naredbe koje se pokreću su:

```
#!/bin/sh
# montiranje
sudo modprobe nbd max_part=16
sudo qemu-nbd -c /dev/nbd0 staza/do/slike
sudo partprobe /dev/nbd0
sudo mount /dev/nbd0p1 staza/montiranja
```

Ispis 2. Montiranje obličja slike *qcow2*

Nakon tog postupka korisniku su dostupne sve datoteke iz operacijskoga sustava TempleOS, te je samo potrebno prekopirati ISO-datoteku distribucije iz staze montiranja na željenu stazu izvan staze montiranja, te obvezno odmontirati montiranu sliku, inače će vrlo vjerojatno doći do gubitka podataka!

```
#!/bin/sh
# odmontiranje
sudo umount staza/montiranja
```

Ispis 3. Odmontiranje obličja slike *raw*

```
#!/bin/sh
# odmontiranje
sudo umount staza/montiranja
sudo qemu-nbd -d /dev/nbd0
sudo partprobe
```

Ispis 4. Odmontiranje obličja slike *qcow2*

5.2. Pokretanje preinačenoga TempleOS-a

Proces instalacije je identičan procesu opisanom u poglavlju 2.1, osim što umjesto službene ISO-slike skida se preinačena slika, dostupna na `https://viktoracoric.xyz/files/TOSHR.ISO.C`.

U svrhu ovoga rada će se rabiti virtualni stroj QEMU, iako bi procedura na drugim virtualnim strojevima kao npr. VirtualBox i VMWare trebala biti dosta čak i jednostavnija.

Prvo je potrebno stvoriti sliku virtualnoga diska. Za to se rabi sljedeća naredba:

```
# stvaranje diska (velicina je opcionalna, velicina u megabajtima se izrazava u
    brojevima bez slova G)
qemu-img create -f qcow2 ~/tos.qcow2 2G
```

Ispis 5. Stvaranje virtualnoga diska

Nakon što je virtualni disk stvoren, treba pokrenuti virtualni stroj.

```
qemu-x86_64 -enable-kvm -cpu host -soundhw pcspk -drive file=tos.qcow2 -cdrom TOSHR.
    ISO.C -m 2G
```

Ispis 6. Pokretanje virtualnoga stroja QEMU u svrhu instalacije putem naredbenoga redka

Argumenti iz naredbe za pokretanje u svrhu instalacije obilježavaju sljedeće:

- `-enable-kvm` znači da se rabi modul Kernel-based Virtual Machine, koji omogućuje sklopovsku virtualizaciju.
- `-soundhw pcspk` je opcionalni argument koji nam omogućuje zvuk unutar TempleOS-a.
- `-cpu host` izbor daje najprecizniju emulaciju za procesor koji korisnik posjeduje.
- `-drive file=[staza]` pruža željenu datoteku virtualnoga diska.
- `-cdrom [staza]` pruža instalacijsku datoteku rabeći pogon CD-ROM.
- `-m [broj]` specificira koliko će radne memorije virtualni stroj rabiti. Potrebno je dati minimalno 512 MB.

Nakon toga slijedi već opisani proces instalacije, koji je sam po sebi jasan, ali ako nije može se podsjetiti u posljednjem odlomku poglavlja 2.1. Operacijski sustav se pokreće sljedećom naredbom:

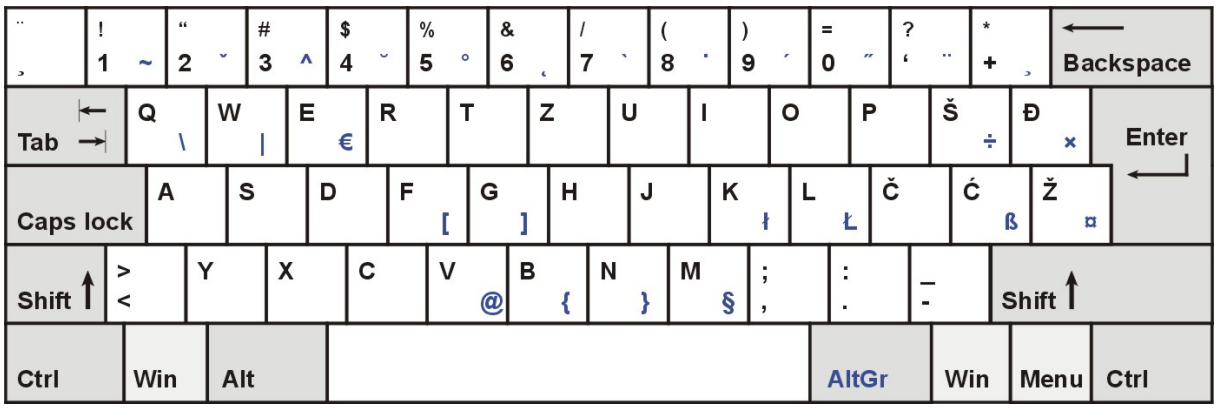
```
qemu-x86_64 -enable-kvm -cpu host -soundhw pcspk -drive file=tos.qcow2 -m 2G
```

Ispis 7. Pokretanje virtualnoga stroja QEMU preko naredbenoga retka

5.3. Ispitivanje preinačenoga TempleOS-a

Nakon što instalacija završi, novoinstalirani operacijski sustav TempleOS se može ugasiti, a iz naredbe iznad se može izbaciti argument `-cdrom`. U novoinstaliranom operacijskom sustavu se može ispitati već iznad opisani proces kojim je hrvatska tipkovnica dodana u sustav: pritisnuti `CTRL-ALT-p` za promjenu tipkovnice na hrvatski jezik, ispitati sve tipke, te po potrebi prebaciti natrag na englesku tipkovnicu.

Model prikazan na Sl. 15 će poslužiti kao referentni model za ispitivanje, jer je hrvatska tipkovnica za TempleOS i rađena po prikazanom modelu.



Slika 15. Referentni model hrvatske tipkovnice

Ispitivanje se vrši pritiskanjem tipaka, ispituju se tipke bez ikakva kombiniranja, tipke u kombinaciji s tipkom `SHIFT`, tipke u kombinaciji s tipkom `ALT`, te tipke u kombinaciji s tipkom `ALT`. Tipke u kombinaciji s tipkom `CTRL` su iste kao i na zadanoj engleskoj tipkovnici, te ih stoga nije nužno ispitati. U radu je već objašnjeno zašto se ne rabe tipke u kombinaciji s tipkama `CTRL` i `ALT`, razlog je taj što operacijski sustav TempleOS već rabi neke funkcije na mjestu gdje bi se inače rabili određeni znakovi.

Kada se završi s ispitivanjem, postoji mogućnost i vratiti se na englesku tipkovnicu ako korisnik to želi.

6. Zaključak

Nakon završene preinake operacijskoga sustava TempleOS, valja se osvrnuti na učinjeno, na sam proces izradbe, na prednosti i nedostatke pristupa opisanih u radu, te prokomentirati procese i metode koje su se ostvarivale u svrhu postizanja cilja.

Instalacija TempleOS-a unutar virtualnoga stroja je prilično jednostavna i cijeli proces je automatiziran. U radu je demonstrirano da se cijeli operacijski sustav instalira za minutu, i to samo uz par pritisaka na tipkovnici. TempleOS nudi i vodič oko sustava, što može puno pomoći korisniku što se tiče osnovne porabe, no za naprjedniju porabu operacijskoga sustava ipak treba imati predznanje iz programiranja. Programski jezik HolyC je vrlo jednostavan za porabu, ali je ujedno i moćan programski jezik koji nudi jednostavan rad s grafikama, dovoljno je jednostavan za podučavanje predškolske djece o programiranju, ali i dovoljno naprjedan da se može isprogramirati TCP/IP stog i pružiti potpora za mrežne operacije. Iz toga se može naslutiti da je dodavanje funkcionalnosti u korisnički prostor, kao i u jezgru, dosta pristupačno, te složenost dodavanja funkcionalnosti ovisi o samoj složenosti te funkcionalnosti.

U radu su se obradila dva načina dodavanja hrvatske tipkovnice, dodavanje unutar korisničkoga prostora i unutar korisničke jezgre. Oba pristupa imaju svoje prednosti i nedostatke. Dodavanje tipkovnice unutar korisničkoga prostora teorijski može biti fleksibilnije, jer iako nije učinjeno u ovom radu, može se ostvariti da se jedan jezik stavi u jednu datoteku. Nedostatak ovoga pristupa je nemogućnost dinamičkoga izmjenjivanja tipkovnice, za koje bi bilo potrebno proširiti jezgru, a bez takve proširke bi trebalo dosta više vremena za obavljanje tako jednostavne operacije. Još jedan nedostatak pristupa dodavanjem unutar korisničkoga prostora jest težina izradbe samog *fonta*. Prednost dodavanja u jezgru operacijskoga sustava je ta što se tipkovnice mogu izmjenjivati dinamički na pritisak tipke. Nedostatak je što kad bi se uključivalo više tipkovnica unutar operacijskoga sustava, operacijski sustav bi rastao u veličini, što je očigledno nešto što je suprotno onomu što je autor TempleOS-a zamislio, budući da je se često hvalio činjenicom da njegov operacijski sustav zauzima svega 2 MB. Još jedan nedostatak je to što je uključivanje fonta i rasporeda tipaka nešto teže.

Također se vrijedi i osvrnuti na metode distribucije učinjenoga unutar operacijskoga sustava TempleOS. U radu je rabljena distribucija cijeloga operacijskog sustava, ali isto tako se može napraviti i instalacijski program koji bi kopirao preinačene datoteke na svoje mjesto. Između ovih dvaju pristupa nema nekih prednosti i nedostataka osim veličine ISO-datoteke, ali se je autor odlučio za prvi pristup jer on ne zahtijeva nikakvo međudjelovanje s korisnikom prigodom prevođenja i pokretanja preinačene inačice operacijskog sustava, dok bi drugi način zahtijevao prevođenje jezgre, koje se vrši Ahead Of Time (AOT), koje zahtijeva korisničku interakciju.

Bilo kako bilo, ovaj rad je pokazao da se potpora za različite tipkovnice može izraditi uz malo truda. Iako je proces dosta jednostavan kada se opiše u radu, proces izradbe je naporniji dio, ali je i iznimno zanimljiv i prigodom izradbe se nauči dosta novih stvari.

Popis literature

[1] T. A. Davis. „The Temple Operating System.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Home/Web/TempleOS.html (pogledano 29. 7. 2021.).

[2] ——, „HolyC.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Doc/HolyC.html (pogledano 3. 8. 2021.).

[3] ——, „DolDoc Overview.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Doc/DolDocOverview.html (pogledano 4. 8. 2021.).

[4] ——, „RedSea File System.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Doc/RedSea.html (pogledano 3. 8. 2021.).

[5] A. S. Tannenbaum i A. S. Woodhull, *Operating Systems: Design and Implementation*. Prentice Hall, 1997.

[6] T. A. Davis. „Block Chain.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Doc/BlkChain.html (pogledano 3. 8. 2021.).

[7] ——, „RedSea Reliability.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Doc/Reliability.html (pogledano 3. 8. 2021.).

[8] ——, „PreProcessor.” (2013.), adresa: https://templeos.holyc.xyz/Wb/Doc/PreProcessor.html (pogledano 24. 8. 2021.).

[9] „QEMU - Arch Wiki.” (2012.), adresa: https://wiki.archlinux.org/title/QEMU#Mounting_a_partition_of_the_guest_on_the_host (pogledano 12. 8. 2021.).

Popis slika