IT4KT

Kód ITMS projektu: 26220220123

Agentúra Ministerstva školstva, vedy, výskumu a športu SR pre štrukturálne fondy EÚ

Výstup: V2.2.5 ImplKomOver-Overenie transferu znalostí

ImplKomIT - Ad-hoc integračný komponent komponent predmetov z oblasti vied informačných a komunikačných technológií

Systém na preberanie a vyhodnocovanie zadaní – programov
Systémová príručka

Riešiteľský kolektív	Genči Ján

Obsah

1. Funkcia programu

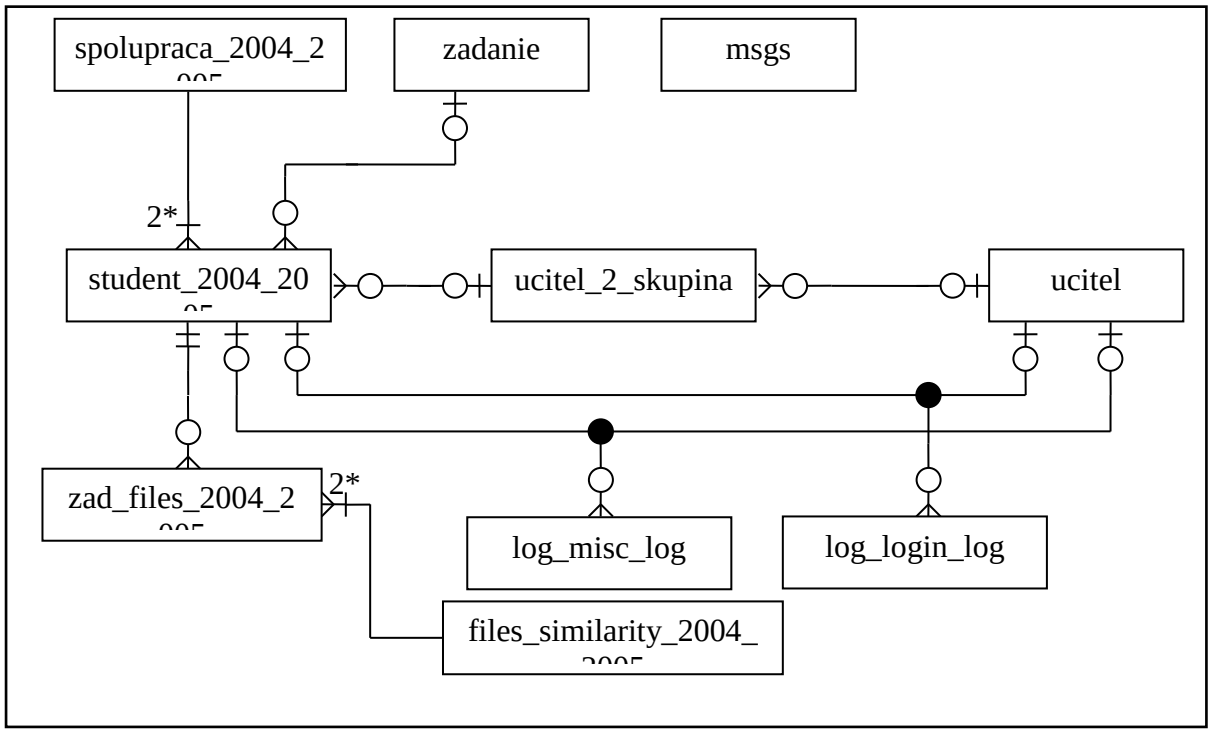
Účel systému je plne automatizovaná, jednoduchá, rýchla, spoľahlivá a objektívna kontrola programových zadaní. Princíp fungovania bude vysvetlený na zadaniach týkajúcich sa súborového systému FAT12/16/32 z predmetu operačné systémy.

2. Popis programu

Systém pracuje pod operačným systémom Linux. Je riešený ako súbor C/C++ programov, shell skriptov a WWW rozhrania, ktoré plní integračnú funkciu. Jadro tvoria C++ programy pracujúce s emulovanými súborovými systémami. Zabezpečujú ich prípravu (vytvorenie emul. súborového systému pred kontrolou zadania, príprava prostredia) a potom kontrolu činnosti zadania. Ďalej určujú podobnosť odovzdaných súborov a riadia front kontroly zadaní. Shell skripty riadia priebeh kontroly zadania a integrujú C/C++ programy s WWW rozhraním. WWW rozhranie je napísané v jazyku PHP s využitím databázového servera MySQL. Zabezpečuje správu používateľov systému a prehľadný prístup k všetkým jeho funkciám.

2.1. Popis databázy

2.1.1. Entitno-relačný diagram



Obr. 2.1 – Entitno-relačný diagram databázy systému

Poznámka: Vzťah medzi dvoma entitami označený symbolom „2*“ znamená, že inštancia entity A sa vzťahuje na práve dve inštancie entity B.

2.1.2. Tabuľka files_similarity_2004_2005

Tabuľka uchováva podobnosť dvojíc už porovnaných súborov. Pre každý akad. rok sa vytvára nová. Súčasťou názvu tabuľky je aj akad. rok, pre ktorý platí.

Stĺpec	Popis	Typ Kľúč
id1	Identifikátor prvého súboru z dvojice (odkaz na stĺpec id tabuľky odovzdaných súborov, *zad_files_2004_2005*)	I P
id2	Identifikátor druhého súboru z dvojice (odkaz na stĺpec id tabuľky odovzdaných súborov, *zad_files_2004_2005*)	I P
similarity	Podobnosť dvojice súborov (v percentách, desatinné číslo)	F -
vlastnik1	Vlastník prvého súboru z dvojice. Identifikátor študenta. Odkaz na stĺpec id tabuľky študentov (*student_2004_2005*).	I -
vlastnik2	Vlastník druhého súboru z dvojice. Identifikátor študenta. Odkaz na stĺpec id tabuľky študentov (*student_2004_2005*).	I -

Tab. 2.1 – Stĺpce tabuľky files_similarity_2004_2005

2.1.3. Tabuľka log_login_log

Systém používa tabuľky s predponou „log_“ v názve s takouto štruktúrou namiesto log súborov. Zaznamenáva do nich rôzne udalosti. Táto tabuľka obsahuje informácie o prihlásení a odhlásení používateľov systému. Na začiatku akad. roku sa jej obsah vymaže.

Stĺpec	Popis	Typ Kľúč
date	Dátum udalosti.	D -
time	Čas udalosti.	T -
sid	Identifikátor prihláseného používateľa („session-id“).	C(100) -
ip	IP adresa používateľa.	C(20) -
host	Názov počítača používateľa.	C(100) -
uid	Identifikátor používateľa (stĺpec id v tabuľke učiteľov alebo študentov, podľa typu používateľa).	I -
priv	Typ používateľa.	C(20) -
msg	Správa.	C(100) -

Tab. 2.2 – Stĺpce tabuľky log_login_log

2.1.4. Tabuľka log_misc_log

Systém používa tabuľky s predponou „log_“ v názve s takouto štruktúrou namiesto log súborov. Zaznamenáva do nich rôzne udalosti. Táto tabuľka obsahuje informácie o rôznych významnejších udalostiach v systéme. Na začiatku akad. roku sa jej obsah vymaže.

Stĺpec	Popis	Typ Kľúč
date	Dátum udalosti.	D -
time	Čas udalosti.	T -
sid	Identifikátor prihláseného používateľa („session-id“).	C(100) -
ip	IP adresa používateľa.	C(20) -
host	Názov počítača používateľa.	C(100) -
uid	Identifikátor používateľa (stĺpec id v tabuľke učiteľov alebo študentov, podľa typu používateľa).	I -
priv	Typ používateľa.	C(20) -
msg	Správa.	C(100) -

Tab. 2.3 – Stĺpce tabuľky log_misc_log

2.1.5. Tabuľka msgs

Obsahuje textové správy, ktoré systém zobrazuje na úvodnej stránke pod prihlasovacím formulárom.

Stĺpec	Popis	Typ Kľúč
id	Identifikátor správy.	I P
msg	Text správy.	TXT -

Tab. 2.4 – Stĺpce tabuľky msgs

2.1.6. Tabuľka spolupraca_2004_2005

Táto tabuľka obsahuje informácie o spolupracujúcich dvojiciach študentov. Platí len pre jeden akad. rok (je súčasťou názvu), na začiatku každého akad. roku sa vytvorí nová.

Stĺpec	Popis	Typ Kľúč
id_stud	Identifikátor študenta. Odkaz na stĺpec id tabuľky študentov (*student_2004_2005*).	I P
id_spoluprac	Identifikátor spolupracovníka študenta. Odkaz na stĺpec id tabuľky študentov (*student_2004_2005*).	I P
spoloc_subory	Podiel spoločných súborov spolupracovníkov (v percentách).	F -

Tab. 2.5 – Stĺpce tabuľky spolupraca_2004_2005

2.1.7. Tabuľka student_2004_2005

Uchováva študentov. Platí len pre jeden akad. rok (je súčasťou názvu), na začiatku každého akad. roku sa vytvorí nová.

Stĺpec	Popis	Typ Kľúč
id	Identifikátor študenta.	I P
id_stud	Identifikačné číslo pridelené školou (TE....).	C(10) -
meno	Meno študenta.	C(20) -
priezvisko	Priezvisko študenta.	C(30) -
tema_zad	Identifikátor zadania. Odkaz na stĺpec ID tabuľky *zadanie*.	I -
skupina	Číslo študijnej skupiny, do ktorej študent patrí. Odkaz na stĺpec id_skup tabuľky *ucitel_2_skupina*.	I -
termin_zad	Najneskorší prípustný termín na odovzdanie zadania.	D -
file_zad	Názov naposledy odoslaného archívu so zadaním na server.	C(200) -
dir_log	Cesta do adresára s výsledkami poslednej kontroly.	C(200) -
stav_zad	Stav zadania. Možné hodnoty: • 0 – nepridelené • 1 – pridelené, žiadny pokus o odovzdanie • 2 – aspoň raz odoslané na kontrolu, zatiaľ neprešlo kontrolou na 100% • 3 – úspešne odovzdané	I -
hodnotenie	Učiteľ má možnosť zapísať si ku každému študentovi textové poznámky. Ľubovoľný text.	TXT -
login	Prihlasovacie meno študenta.	C(35) UNQ
pwd	Heslo študenta zašifrované funkciou MD5 (v prípade internej autentifikácie). Ak tento stĺpec obsahuje reťazec „LDAP“, signalizuje to autentifikáciu cez LDAP server.	C(40) -
meno_ucitela	Meno učiteľa, ktorý učí študenta. Pridelené na základe príslušnosti študenta do určitej študijnej skupiny.	C(200) -
nazov_zad	Názov zadania.	C(100) -
datum_odovzd	Dátum a čas posledného odoslania zadania na kontrolu.	DTM -
pocet_odovzd	Celkový počet pokusov o odovzdanie.	I -
results_checked	Príznak. Indikuje, či si už študent pozrel výsledky poslednej kontroly zadania.	I -
orig_subory	Podiel originálnych súborov odoslaných študentom (v percentách).	F -

Tab. 2.6 – Stĺpce tabuľky student_2004_2005

2.1.8. Tabuľka zad_files_2004_2005

Obsahuje informácie o súboroch odoslaných študentmi na kontrolu. Neobsahuje súbory typu makefile. Platí len pre jeden akad. rok (je súčasťou názvu), na začiatku každého akad. roku sa vytvorí nová.

Stĺpec	Popis	Typ Kľúč
id	Identifikátor súboru.	I P
tstamp	Dátum a čas súboru vo forme „časovej pečiatky“.	I -
cislo_k	Poradie kontroly.	I -
archiv	Názov súboru s archívom, ktorý obsahuje príslušný súbor.	C(255) -
nazov	Názov samotného súboru.	C(255)
vlastnik	Vlastník súboru. Odkaz na stĺpec id tabuľky študentov (*student_2004_2005*).	I -
velkost	Veľkosť súboru v bajtoch.	I -
hash	Hash súboru vypočítaný funkciou MD5.	C(50)

Tab. 2.7 – Stĺpce tabuľky zad_files_2004_2005

2.1.9. Tabuľka zadanie

Obsahuje zadania prítomné v systéme.

Stĺpec	Popis	Typ Kľúč
id	Identifikátor zadania.	I P
nazov	Názov zadania.	C(100) -
popis	Plné znenie zadania. Môže byť vo formáte HTML, max. 64KB.	TXT -
skript	Skript, ktorý zabezpečí kontrolu zadania. Systém tento skript spustí po odoslaní archívu so zadaním. Skript sa musí nachádzať v adresári danom konštantou **SCRIPTPATH** definovanou v súbore **path_cfg.php** (kap. 2.5.27).	C(50) -

Tab. 2.8 – Stĺpce tabuľky zadanie

2.1.10. Tabuľka ucitel

Uchováva informácie o učiteľoch.

Stĺpec	Popis	Typ Kľúč
id	Identifikátor učiteľa.	I P

meno	Meno učiteľa.	C(20) -
priezvisko	Priezvisko učiteľa.	C(30) -
login	Prihlasovacie meno učiteľa.	C(35) UNQ
pwd	Heslo šifrované funkciou MD5.	C(40) -
garant	Príznak garanta. Ak je 1, učiteľ je tzv. garant, teda má prístup k všetkým študentom bez ohľadu na študijnú skupinu.	I -

Tab. 2.9 – Stĺpce tabuľky ucitel

2.1.11. Tabuľka ucitel_2_skupina

Určuje príslušnosť študijných skupín k učiteľom.

Stĺpec	Popis	Typ Kľúč
id_uc	Identifikátor učiteľa.	I P
id_skup	Identifikátor skupiny.	I P

Tab. 2.10 – Stĺpce tabuľky ucitel_2_skupina

2.1.12. Vysvetlivky k typom a kľúčom stĺpcov

Typ alebo kľúč	Popis
I	32-bitové celé číslo.
P	Primárny kľúč.
C(x)	Reťazec dlhý maximálne x znakov.
D	Dátum.
F	Číslo s pohyblivou rádovou čiarkou (32 bitov).
TXT	Text. Maximálne 64 KB.
T	Čas.
DTM	Dátum a čas.
UNQ	Unikátne hodnoty.

Tab. 2.11 – Vysvetlivky k typom a kľúčom stĺpcov

2.2. Popis funkcie a štruktúry C/C++ programov

2.2.1. efs_content

Program vypisujúci obsah emulovaného súborového systému (všetky súbory a adresáre) do HTML súboru EFSContent_log.html.

Použité moduly: EFATFS, fatlib, LogSystem, xcompat

Spustenie a argumenty príkazového riadku:
`efs_content efs`

Argument	Popis
`efs_content`	Názov programu
`efs`	Povinný argument. Názov súboru s emulovaným súborovým systémom. Názov musí obsahovať jeden z nasledujúcich reťazcov: f12, F12, F16, f16, F32, f32. Program tieto reťazce využíva na rozpoznanie typu EFS, napríklad súbor FAT.F12 bude pokladať za emulovaný súborový systém typu FAT12.

Tab. 2.12 – Argumenty príkazového riadku programu efs_content

Návratové hodnoty:

Hodnota	Význam
0	Úspešné vykonanie programu.
1	Počas vykonávania programu došlo k niektorej z nasledujúcich chýb: - nebol zadaný požadovaný argument príkazového riadku - nepodarilo sa určiť typ FAT podľa zadaného argumentu - nepodarilo sa inicializovať emulovaný FS (napr. súbor s EFS nemožno otvoriť, súbor nie je FATXX, apod.)

Tab. 2.13 – Návratové hodnoty programu efs_content

2.2.2. efs2rfs_prep_f12

Program slúži na prípravu prostredia na spustenie a kontrolu zadania čítajúceho súbory z emulovaného súborového systému FAT12. Ak v aktuálnom adresári existuje súbor **time_cfg**, program ho spracuje takto:

- prvý riadok musí byť v tvare „`time chrt`“ (pochopiteľne bez úvodzoviek), kde:
  - `time` – maximálny čas behu zadania v sekundách
  - `chrt` – reťazec, prípustné hodnoty sú:
    - chroot – program vygeneruje do výsledného skriptu reťazec „`sudo /usr/sbin/chroot `pwd``“ pred volanie programu runner – ide o spustenie príkazu chroot (vytvorenie oddeleného filesystému), pričom utilita sudo musí byť nastavená tak, aby volanie chroot umožnila bez hesla pre príslušného používateľa.
    - nochroot – program vygeneruje skript, ktorý bude volať priamo runner bez volaní sudo a chroot.
- za parametrom `chrt` môžu nasledovať dodatočné parametre pre program runner oddelené medzerou.

Výsledkom činnosti tohto programu je (v aktuálnom adresári):

- adresár work (prázdny) – do tohto adresára musí zadanie zapísať súbory prečítané z emulovaného súborového systému

- súbor EFSprep_log.html – obsahuje použité šablóny, vygenerované mená súborov, obsah emulovaného filesystému, plánované spustenie zadania a presmerovanie jeho výstupov.
- FATwork.f12 – vytvorený pracovný emulovaný súborový systém
- súbor _to_check – zoznam súborov, ktoré *musí* zadanie skopírovať z emul. súborového systému do adresára work v reálnom súborovom systéme (na každom riadku je jeden súbor)
- súbor _to_check_2 – zoznam súborov, ktoré zadanie *nesmie* skopírovať do reálneho súborového systému (na každom riadku je jeden súbor)
- súbor _to_run – skript zabezpečujúci spustenie zadania so správnymi parametrami, tiež ošetruje obmedzenie dostupných systémových prostriedkov pre zadanie využitím programu runner

Poznámka: Je potrebné zabezpečiť, aby zadanie nemalo prístup k súborom _to_check a _to_check_2. Zabezpečuje to skript riadiaci priebeh kontroly (kap. 2.4.1).

Použité moduly: EFATFS, fatlib, LogSystem, xcompat

Spustenie a argumenty príkazového riadku:
`efs2rfs_prep_f12 [path]`

Argument	Popis
`efs2rfs_prep_f12`	Názov programu
`path`	Nepovinný argument. Reťazec, ktorý bude pripojený pred volanie programu runner vo vygenerovanom skripte _to_run (použiteľné ako cesta k programu runner)

Tab. 2.14 – Argumenty príkazového riadku programu efs2rfs_prep_f12

Návratové hodnoty:

Hodnota	Význam
Nedefinovaná	Nedefinovaný

Tab. 2.15 – Návratové hodnoty programu efs2rfs_prep_f12

2.2.3. efs2rfs_prep_f16, efs2rfs_prep_f32

Analogické programy ako efs2rfs_prep_f12, ale zabezpečujú generovanie emulovaného súborového systému typu FAT16/32. Názvy vstupných a výstupných súborov sú totožné okrem súboru s emulovaným súborovým systémom (FATwork.f16 alebo FATwork.f32 podľa typu FAT). Argumenty príkazového riadku a návratové hodnoty sú takisto identické.

2.2.4. efs2rfs_check_f12

Program na kontrolu stavu emulovaného a reálneho súborového systému po spustení zadania. Používa súbory FATwork.f12, _to_check a _to_check_2 vytvorené programom efs2rfs_prep_f12. Uvedené súbory musia byť v aktuálnom adresári. Výsledkom činnosti

programu je súbor EFScheck_log.html (v aktuálnom adresári), ktorý obsahuje kontrolované položky, výsledok kontroly a percentuálne hodnotenie úspešnosti zadania. Program overí *existenciu* a obsah súborov uvedených v súbore _to_check a *neexistenciu* súborov uvedených v súbore _to_check_2. Celkový výsledok kontroly program indikuje návratovou hodnotou.

Použité moduly: EFATFS, fatlib, LogSystem, xcompat

Spustenie a argumenty príkazového riadku:
`efs2rfs_check_f12`

Argument	Popis
`efs2rfs_check_f12`	Názov programu

Tab. 2.16 – Argumenty príkazového riadku programu efs2rfs_check_f12

Návratové hodnoty:

Hodnota	Význam
0	Úspešné vykonanie programu a kontroly. Úloha stanovená zadaním splnená na 100%.
1	Úloha stanovená zadaním nebola splnená na 100% alebo došlo k inej chybe.

Tab. 2.17 – Návratové hodnoty programu efs2rfs_check_f12

2.2.5. efs2rfs_check_f16, efs2rfs_check_f32

Analogické programy ako efs2rfs_check_f12, ale zabezpečujú kontrolu zadania pre emulovaný súborový systému typu FAT16/32. Názvy vstupných a výstupných súborov sú totožné okrem súboru s emulovaným súborovým systémom (FATwork.f16 alebo FATwork.f32 podľa typu FAT). Argumenty príkazového riadku a návratové hodnoty sú takisto identické.

2.2.6. rfs2efs_prep_f12

Program slúži na prípravu prostredia na spustenie a kontrolu zadania zapisujúceho súbory z reálneho súborového systému do emulovaného súborového systému FAT12 (zadanie popísané v kap. 2. 3. 1., program realizuje body b-f podľa kap. 2. 4. 1.). Ak v aktuálnom adresári existuje súbor **time_cfg**, program ho spracuje takto:

- prvý riadok musí byť v tvare „`time chrt`“ (pochopiteľne bez úvodzoviek), kde:
  - `time` – maximálny čas behu zadania v sekundách
  - `chrt` – reťazec, prípustné hodnoty sú:
    - chroot – program vygeneruje do výsledného skriptu reťazec „`sudo /usr/sbin/chroot \`pwd\``“ pred volanie programu runner – ide o spustenie príkazu chroot (vytvorenie oddeleného filesystému), pričom utilita sudo musí byť nastavená tak, aby volanie chroot umožnila bez hesla pre príslušného používateľa.
    - nochroot – program vygeneruje skript, ktorý bude volať priamo runner bez volaní sudo a chroot.

- za parametrom `chrt` môžu nasledovať dodatočné parametre pre program runner oddelené medzerou.

Výsledkom činnosti tohto programu je (v aktuálnom adresári):

- adresár work – obsahuje súbory, z ktorých *niektoré* (vyhovujúce príslušnej šablóne) musí zadanie zapísať do emulovaného súborového systému
- adresár work_z – kópia adresára work (pre prípad pokusu zadania o manipuláciu so súbormi v adresári work)
- súbor EFSprep_log.html – obsahuje použité šablóny, vygenerované mená súborov, obsah emulovaného filesystému, plánované spustenie zadania a presmerovanie jeho výstupov.
- FATwork.f12 – vytvorený pracovný emulovaný súborový systém
- súbor _to_check – zoznam súborov, ktoré *musí* zadanie skopírovať z adresára work do emul. súborového systému (na každom riadku je jeden súbor)
- súbor _to_check_2 – zoznam súborov, ktoré zadanie *nesmie* skopírovať z adresára work do emul. súborového systému (na každom riadku je jeden súbor)
- súbor _to_run – skript zabezpečujúci spustenie zadania so správnymi parametrami, tiež ošetruje obmedzenie dostupných systémových prostriedkov pre zadanie využitím programu runner

Poznámka: Je potrebné zabezpečiť, aby zadanie nemalo prístup k súborom _to_check a _to_check_2. Zabezpečuje to skript riadiaci priebeh kontroly (kap. 2.4.1).

Použité moduly: EFATFS, fatlib, LogSystem, xcompat

Spustenie a argumenty príkazového riadku:
`rfs2efs_prep_f12 [path]`

Argument	Popis
`rfs2efs_prep_f12`	Názov programu
`path`	Nepovinný argument. Reťazec, ktorý bude pripojený pred volanie programu runner vo vygenerovanom skripte _to_run (použiteľné ako cesta k programu runner)

Tab. 2.18 – Argumenty príkazového riadku programu rfs2efs_prep_f12

Návratové hodnoty:

Hodnota	Význam
Nedefinovaná	Nedefinovaný

Tab. 2.19 – Návratové hodnoty programu rfs2efs_prep_f12

2.2.7. rfs2efs_prep_f16, rfs2efs_prep_f32

Analogické programy ako rfs2efs_prep_f12, ale zabezpečujú generovanie emulovaného súborového systému typu FAT16/32. Názvy vstupných a výstupných súborov sú totožné okrem súboru s emulovaným súborovým systémom (FATwork.f16 alebo FATwork.f32 podľa typu FAT). Argumenty príkazového riadku a návratové hodnoty sú takisto identické.

2.2.8. rfs2efs_check_f12

Program na kontrolu stavu emulovaného a reálneho súborového systému po spustení zadania. Používa súbory FATwork.f12, _to_check a _to_check_2 vytvorené programom rfs2efs_prep_f12. Uvedené súbory musia byť v aktuálnom adresári. Výsledkom činnosti programu je súbor EFScheck_log.html (v aktuálnom adresári), ktorý obsahuje kontrolované položky, výsledok kontroly a percentuálne hodnotenie úspešnosti zadania. Program overí *existenciu* a obsah súborov uvedených v súbore _to_check a *neexistenciu* súborov uvedených v súbore _to_check_2. Celkový výsledok kontroly program indikuje návratovou hodnotou.

Použité moduly: EFATFS, fatlib, LogSystem, xcompat

Spustenie a argumenty príkazového riadku:
`rfs2efs_check_f12`

Argument	Popis
`rfs2efs_check_f12`	Názov programu

Tab. 2.20 – Argumenty príkazového riadku programu rfs2efs_check_f12

Návratové hodnoty:

Hodnota	Význam
0	Úspešné vykonanie programu a kontroly. Úloha stanovená zadaním splnená na 100%.
1	Úloha stanovená zadaním nebola splnená na 100% alebo došlo k inej chybe.

Tab. 2.21 – Návratové hodnoty programu rfs2efs_check_f12

2.2.9. rfs2efs_check_f16, rfs2efs_check_f32

Analogické programy ako rfs2efs_check_f12, ale zabezpečujú kontrolu zadania pre emulovaný súborový systému typu FAT16/32. Názvy vstupných a výstupných súborov sú totožné okrem súboru s emulovaným súborovým systémom (FATwork.f16 alebo FATwork.f32 podľa typu FAT). Argumenty príkazového riadku a návratové hodnoty sú takisto identické.

Poznámka: Pokiaľ zadanie poškodí emulovaný súborový systém nesprávnym zápisom, môže sa stať, že kontrolný program spadne. Túto situáciu ošetruje skript riadiaci priebeh kontroly zadania.

2.2.10. rfs2efs_prep_f12_shnames, rfs2efs_prep_f16_shnames, rfs2efs_prep_f32_shnames

Programy na prípravu prostredia na kontrolu činnosti zadania zapisujúceho súbory do

emulovaného súborového systému bez použitia dlhých názvov. Programy sú úplne analogické ako rfs2efs_prep_fxx. Kontrola sa realizuje (takisto ako pri kontrole zadaní používajúcich dlhé mená) pomocou programov rfs2efs_check_fxx.

2.2.11. ldd_pars

Jednoduchý program, ktorý číta zo štandardného vstupu výstup programu ldd a generuje na štandardný výstup skript zabezpečujúci skopírovanie potrebných dynamických knižníc. Používa sa pri vytváraní nového koreňového adresára pre oddelený filesystém.

Príklad:

```
[mx@pocitac1 mx]$ ldd FATprogs/efs2rfs_prep_f12

        linux-gate.so.1 =>  (0xffffe000)

        libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0x40024000)

        libm.so.6 => /lib/tls/libm.so.6 (0x400f6000)

        libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x40119000)

        libc.so.6 => /lib/tls/libc.so.6 (0x40123000)

        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)


[mx@pocitac1 mx]$ ldd FATprogs/efs2rfs_prep_f12 | FATprogs/ldd_pars lib

cp -f /usr/lib/libstdc++.so.6 lib

cp -f /lib/tls/libm.so.6 lib

cp -f /lib/libgcc_s.so.1 lib

cp -f /lib/tls/libc.so.6 lib

cp -f /lib/ld-linux.so.2 lib
```

Spustenie a argumenty príkazového riadku:

```
ldd_pars destdir
```

Argument	Popis
ldd_pars	Názov programu
destdir	Povinný argument. Cieľový adresár na skopírovanie knižníc.

Tab. 2.22 – Argumenty príkazového riadku programu ldd_pars

Návratové hodnoty:

Hodnota	Význam
0	Úspešné vykonanie programu.

Tab. 2.23 – Návratové hodnoty programu ldd_pars

2.2.12. make_test

Program na testovanie príkazov vykonaných utilitou make počas kompilácie zadania. Kompilácia neprebieha v oddelenom filesystéme a makefile umožňuje vykonať ľubovoľný príkaz. Program pracuje tak, že číta zo štandardného vstupu výstup utility make s parametrom –n (make len vypíše vykonávané príkazy, ale nespustí ich). Pokiaľ by došlo k spusteniu iného príkazu ako „gcc *“, „cc *“ alebo „g++ *“, program to vyhodnotí ako nepovolený príkaz (* predstavuje ľubovoľnú postupnosť znakov okrem znakov „|“ a „;“). Návratová hodnota indikuje výsledok testu.

Spustenie a argumenty príkazového riadku:

```
make_test
```

Argument	Popis
`make_test`	Názov programu

Tab. 2.24 – Argumenty príkazového riadku programu make_test

Návratové hodnoty:

Hodnota	Význam
0	Úspešné vykonanie programu. Žiadne nepovolené príkazy neboli detekované.
1	Kompilácia zadania by viedla k vykonaniu nepovolených príkazov (iných ako volanie kompilátora).

Tab. 2.25 – Návratové hodnoty programu make_test

2.2.13. ngramcmp

Program na určenie podobnosti (textových) súborov na základe n-gramov. Pri porovnaní dvoch súborov program vypíše na štandardný výstup podobnosť súborov v percentách. V prípade porovnávania viacerých súborov (prepínač –i) program takisto vypisuje na štandardný výstup podobnosti jednotlivých dvojíc porovnávaných súborov, jeden riadok výstupu zodpovedá jednej porovnanej dvojici. Ak sa použije prepínač –v, program vypisuje viac informácií o porovnávaných súboroch.

Spustenie a argumenty príkazového riadku:

```
ngramcmp [-c] [-h] subor1 subor2
ngramcmp [-c] [-h] -i
```

Argument	Popis
`ngramcmp`	Názov programu
`-c`	Uloženie n-gramového profilu na disk. Profil každého porovnávaného súboru bude uložený na disk do súboru *nazov_porovnavaneho_suboru.ngram-profile*. Pokiaľ už takýto profil

	pre súbor existuje, program ho načíta z uvedeného súboru a nepočíta ho znovu. Užitočné pri vzájomnom porovnávaní väčšieho množstva súborov, predovšetkým v prípade porovnávania jedného súboru s viacerými rôznymi súbormi.
`-h`	Pri výpočte profilu sa použije hashovacia tabuľka namiesto štandardných C++ stromov. Podstatne rýchlejšia metóda ako stromy.
`subor1, subor2`	Súbory na porovnanie.
`-i`	Porovnanie viacerých dvojíc súborov. Program číta štandardný vstup. Na každom riadku očakáva dvojicu súborov na porovnanie oddelenú medzerou alebo tabulátorom. Tento prepínač je užitočný v kombinácii s –c.

Tab. 2.26 – Argumenty príkazového riadku programu ngramcmp

Poznámky: Prepínače možno uvádzať v ľubovoľnom poradí, vždy však pred porovnávanými súbormi. Možno ich uvádzať spolu aj oddelene (-ch alebo –c –h). Vždy sa odporúča použiť prepínač –h.

Návratové hodnoty:

Hodnota	Význam
0	Úspešné porovnanie dvojice súborov (bez prepínača –i). V prípade použitia tohto prepínača je návratová hodnota **vždy** 0.
1	V prípade porovnania dvojice súborov došlo k chybe: - aspoň jeden zo vstupných súborov nemožno otvoriť - preplnenie buffra tokenizéra (napr. v prípade pokusu o porovnanie binárnych súborov) - nesprávne použitie argumentov príkazového riadku

Tab. 2.27 – Návratové hodnoty programu ngramcmp

2.2.14. sblock

Skript riadiaci kontrolu zadania spúšťa tento program. Jeho úlohou je pozastaviť vykonávanie kontroly (volajúceho procesu) v prípade, ak už prebieha iná kontrola. Využíva semafor.

Spustenie a argumenty príkazového riadku:

```
sblock cesta op
```

Argument	Popis
`sblock`	Názov programu
`cesta`	Cesta (len cesta, nie názov súboru) k súboru s kľúčom semafora, program musí mať právo zapisovať do tohto adresára.
`op`	Jedna z nasledujúcich operácií: • + blokovanie semafora. Predpokladom je existujúci súbor *smky* (kľúč semafora) v adresári uvedenom v argumente *cesta* a

	existujúci prístupný semafor s takýmto kľúčom, alebo ak súbor *smky* neexistuje, program sa pokúsi vytvoriť nový semafor s hodnotou 1 a jeho kľúč uložiť do tohto súboru. Pokiaľ nie je splnený niektorý z týchto predpokladov, program skončí s návratovou hodnotou 1. Inak zníži hodnotu semafora o 1 a skončí s návratovou hodnotou 0. Pokiaľ semafor nemožno znížiť (urobila to iná inštancia tohto programu), OS program pozastaví (a tým aj volajúci proces – kontrolu), až kým sa hodnota semaforu opäť nezvýši.
	• - odblokovanie semafora. Predpokladom je existujúci súbor *smky* (kľúč semafora) v adresári uvedenom v argumente *cesta* a existujúci prístupný semafor s takýmto kľúčom, alebo ak súbor *smky* neexistuje, program sa pokúsi vytvoriť nový semafor s hodnotou 1 a jeho kľúč uložiť do tohto súboru. Pokiaľ nie je splnený niektorý z týchto predpokladov, program skončí s návratovou hodnotou 1. Inak zvýši hodnotu semafora o 1 a skončí s návratovou hodnotou 0. Pokiaľ semafor nemožno zvýšiť, skončí s návratovou hodnotou 1.
	• t test semafora. Po reštarte servera dôjde k situácii, že súbor *smky* existuje, ale zodpovedajúci semafor nie. Táto operácia detekuje takýto stav. Predpokladom je existujúci súbor *smky* (kľúč semafora) v adresári uvedenom v argumente *cesta* a existujúci prístupný semafor s takýmto kľúčom, alebo ak súbor *smky* neexistuje, program sa pokúsi vytvoriť nový semafor s hodnotou 1 a jeho kľúč uložiť do tohto súboru. Pokiaľ nie je splnený niektorý z týchto predpokladov, program skončí s návratovou hodnotou 1, inak 0.
	• r oprava semafora. Ak dôjde k situácii, že súbor *smky* existuje, ale zodpovedajúci semafor nie, program sa pokúsi túto situáciu vyriešiť:
	  • zmaže obsah adresára uvedeného v argumente *cesta* (v tomto adresári môžu byť nedokončené kontroly z dôvodu pádu/reštartu servera)
	  • vytvorí nový semafor a súbor *smky*

Tab. 2.28 – Argumenty príkazového riadku programu sblock

Návratové hodnoty:

Hodnota	Význam
0	Úspešné vykonanie požadovanej operácie nad semaforom.
1	Chyba vykonania požadovanej operácie nad semaforom, pravdepodobne spôsobená reštartom servera. Odporúča sa použiť operáciu r. Ak aj táto vráti 1, nemožno vytvoriť nový semafor s unikátnym kľúčom alebo nemožno zapísať súbor *smky* do adresára uvedeného v argumente *cesta*.

Tab. 2.29 – Návratové hodnoty programu sblock

2.2.15. runner

Tento program slúži na spustenie ľubovoľného iného programu. Umožňuje obmedziť maximálny čas behu spúšťaného programu, presmerovať jeho výstupy a obmedziť dostupné systémové prostriedky (čas procesora, adresný priestor, maximálnu veľkosť súboru vytvoreného programom).

Spustenie a argumenty príkazového riadku:
`runner [-r vystup ch_vystup] [-m mlimit] [-f flimit] [-t tlimit] cas program [par1 par2 ... parn]`

Argument	Popis
`runner`	Názov programu
`-r`	Príznak presmerovania výstupov spúšťaného programu.
`vystup`	Názov súboru, do ktorého bude presmerovaný štandardný výstup spúšťaného programu.
`ch_vystup`	Názov súboru, do ktorého bude presmerovaný chybový výstup spúšťaného programu.
`-m`	Príznak obmedzenia maximálnej veľkosti adresného priestoru spúšťaného programu.
`mlimit`	Obmedzenie maximálnej veľkosti adresného priestoru spúšťaného programu v bajtoch.
`-f`	Príznak obmedzenia maximálnej veľkosti súborov vytvorených spúšťaným programom.
`flimit`	Obmedzenie maximálnej veľkosti súborov vytvorených spúšťaným programom v bajtoch.
`-t`	Príznak obmedzenia času procesora pre spúšťaný program.
`tlimit`	Obmedzenie času procesora pre spúšťaný program v sekundách.
`cas`	Maximálny čas behu (bez ohľadu na využitie procesora) spusteného programu v sekundách. Runner overuje tento čas každú sekundu. Ak program beží dlhšie než povolený čas, runner použije kill.
`program`	Súbor, ktorý bude spustený.
`par1 ... parn`	Argumenty príkazového riadku, ktoré runner odovzdá spustenému programu.

Tab. 2.30 – Argumenty príkazového riadku programu runner

Návratové hodnoty:

Hodnota	Význam
128	Chyba argumentov príkazového riadku.
129	Chyba pri spustení cieľového programu, chyba volaní fork/exec.
130	Cieľový program bežal dlhšie než stanovený limit (argument *cas*), bol použitý kill.
131	Cieľový program sa neskončil korektne (ukončil ho nejaký signál, viac informácií o príčine ukončenia programu vypíše runner na štandardný

	výstup).
132	Chyba pri pokuse o nastavenie limitu pre niektorý zo systémových prostriedkov (chyba setrlimit).
Iná	V prípade úspešného spustenia a ukončenia cieľového programu sa návratová hodnota zhoduje s návratovou hodnotou vrátenou cieľovým programom.

Tab. 2.31 – Návratové hodnoty programu runner

2.3. Popis C/C++ modulov, tried a údajových štruktúr

2.3.1. Modul fatlib (súbory fatlib.h, fatlib.cpp)

Tento modul predstavuje jadro celého systému, poskytuje metódy na prácu s emulovaným súborovým systémom.

Údajové štruktúry
Údajové štruktúry použité v tomto module nebudem bližšie popisovať, ich zmysel by mal byť jasný – vyplýva zo štruktúry FAT filesystémov. Detailný popis je uvedený v [1] alebo [2].
Štruktúra *NonPortableLongDirectoryEntry* je použitá z dôvodu nekompatibility typu *wchar_t* pod OS Windows (16 bitov) a OS Linux (32 bitov), pričom unikódový znak (použitý v dlhých menách súborového systému FAT) má len 16 bitov.

Popis tried

Trieda *FATFileSystem* – dátové členy	
Deklarácia	**Popis**
`DirectoryEntry de;`	Položka adresára
`FILE* f;`	Smerník na súbor s emulovaným súborovým systémom
`long RootPos;`	Absolútna pozícia koreňového adresára v súbore s emulovaným FS
`long SystemSectors;`	Počet sektorov pred prvým dátovým sektorom, používa sa pre interné výpočty
`long FirstDataSector;`	Index prvého dátového sektora (hodnota totožná s predošlou premennou)
`long FirstFATSector;`	Index prvého sektora prvej FAT tabuľky
`long DEsPerCluster;`	Počet položiek adresára v jednom clustri (DirectoryEntries Per Cluster)
`char* FATImageName;`	Názov súboru s emulovaným FS
`FATType FAT;`	Typ EFS. Enumeračný typ, prípustné sú hodnoty FAT12(0), FAT16(1) a FAT32(2).
`int InitSucc;`	Indikuje úspešnú inicializáciu EFS.

`FATBootRecord fbr;`	Boot sector - obsahuje všetky charakteristiky EFS.
`bool FATBuffering;`	Príznak indikujúci aktívny „FAT Buffering“ – FAT tabuľka (alebo jej časť) je v pamäti pre zrýchlenie prístupu k jej položkám – veľmi vhodné napr. pri zápise súborov do EFS. Pri FAT12/16 je v pamäti celá FAT tabuľka (jej max. veľkosť je 128 KB). Pri FAT32 je v pamäti len časť, s ktorou sa práve pracuje (FAT32 môže zaberať teoreticky aj niekoľko GB).
`char *FATBuf;`	Smerník na oblasť pamäte, v ktorej je uložená FAT tabuľka (alebo jej časť)
`unsigned FirstBufItem`	Prvá položka FAT tabuľky, ktorá sa už nachádza v buffri (len pre FAT32)
`unsigned NumItems;`	Počet položiek FAT tabuľky, ktoré sa nachádzajú v buffri (len pre FAT32)
`int LastErrNo;`	Premenná obsahujúca číslo poslednej chyby. Zatiaľ sa vzťahuje len na funkciu prečítania súboru z EFS do reálneho súborového systému (funkcia ReadFile(...), používa sa pri diagnostike zadaní).

Tab. 2.32 – Dátové členy triedy FATFileSystem

Trieda *FATFileSystem* – členské funkcie	
Deklarácia	**Popis**
`FATFileSystem(void);`	Konštruktor triedy – alokuje pamäť pre FAT buffer
`~FATFileSystem(void);`	Deštruktor triedy – ak bol aktívny FAT buffering, zapíše buffer na disk. Uvoľní pamäť FAT buffra. Ak bol filesystém inicializovaný, uzavrie súbor s EFS.
`int Init(char *fatimagename, FATType type, bool FATBuff=false);`	Inicializácia EFS. fatimagename – smerník na reťazec obsahujúci názov súboru s emulovaným FS. type – typ EFS (0 – FAT12, 1 – FAT16, 2 – FAT32) FATBuff – nepovinný parameter. Ak je true, aktivuje sa FAT buffering – zrýchľuje prácu s EFS. Implicitná hodnota je false. FAT buffering je vysvetlený v časti „dátové členy“. Funkcia po úspešnej inicializácii vráti nenulovú hodnotu, v prípade chyby 0.
`int ReadBootRecord(char *filename);`	Funckia načíta bootblok EFS a uloží ho do premennej fbr. V prípade chyby vráti 0, inak 1. Netreba ju volať, volá ju funkcia *Init(..)*.
`int PrintDir(char *dir);`	Funkcia na výpis obsahu (súborov a podadresárov) zadaného adresára EFS do textového okna (konzoly). Podporuje dlhé mená súborov. Táto verzia pracovala v konzolovej verzii aplikácie. Funkcia vráti 1, ak sa adresár podarilo nájsť v EFS a vypísať, inak vráti 0.
`long FindDirectoryEntry(char *name, long *DESoffsets=NULL);`	Táto funkcia slúži na vyhľadanie DirectoryEntry zadanej položky (súboru alebo adresára, poprípade aj s cestou). name – meno hľadanej položky (dlhý alebo krátky názov),

	v poli DESoffsets vráti offsety začiatkov LDE, ktoré patria danej položke – používa sa pri mazaní dlhých mien presahujúcich cluster. Všetky položky poľa sú platné až po prvú nulu. **POLE MUSÍ MAŤ ASPOŇ 20 POLOŽIEK** (inak hrozí pád programu). DE s krátkym názvom je načítaná do členskej premennej de triedy. Z nej možno získať napr. informácie o dĺžke súboru, atribútoch atď. (Pomocou funkcie *GetDE()*). Ak sa podarilo položku nájsť, funkcia vráti offset SDE v súbore s EFS, inak vráti 0.
`void PrintInfo(void);`	Funkcia vypíše základné informácie o EFS. Nemá parametre ani návratovú hodnotu.
`unsigned GetFATItem(unsigned n);`	Vráti *n-tú* položku z prvej FAT tabuľky EFS. Funkcia pracuje so všetkými typmi FAT tabuliek (12/16/32).
`void SetFATItem(unsigned n, unsigned value);`	Nastaví n-tú položku vo všetkých FAT tabuľkách v EFS na hodnotu value.
`int DelFile(char *name);`	Zmaže súbor v EFS. Parametrom je meno a popr. cesta k súboru na zmazanie v EFS. V prípade úspešného zmazania vráti 1, inak 0. Podporuje dlhé názvy. Korektné je zmazať aj prázdny adresár. Zmazanie neprázdneho adresára síce prebehne, ale spôsobí poškodenie EFS (stratené clustre).
`void GoToCluster(unsigned cl);`	Nastaví ukazovateľ v súbore emulovaného FS na cluster cl. Funguje pre všetky typy FAT.
`unsigned FreeSpace(void);`	Vráti počet voľných bajtov v EFS. Maximálne však len do 4 GB (32-bitové celé číslo bez znamienka). Funguje pre všetky typy FAT.
`unsigned FreeClusters(void);`	Vráti počet voľných clustrov v EFS. Pracuje pre všetky typy FAT.
`unsigned GetFreeCluster(bool Fragments=false);`	Vráti číslo voľného clustra s najnižším číslom. Ak žiadny cluster nie je voľný, vráti 0. Fragments – príznak fragmentácie, nepovinný parameter. Ak je true, funkcia nevráti voľný cluster s najnižším číslom, ale náhodný voľný cluster. Predpokladá, že všetky clustre od prvého voľného sú voľné (až do konca EFS). Toto spôsobí fragmentáciu súborov. Používa sa pri overení obnovenia zmazaných súborov (obnoviť fragmentovaný súbor znamená zapamätať si clustre, v ktorých bol uložený).
`long GetFreeDEPosition(char *path,unsigned long *parentcluster=NULL);`	Vráti offset v súbore EFS, na ktorý možno uložiť novú položku do zadaného adresára. path – adresár (cesta), v ktorom hľadáme pozíciu pre novú DE (prázdny reťazec pre ROOT). parentcluster – prostredníctvom tohto smerníka (nepovinný parameter) funkcia vráti prvý cluster adresára, v ktorom hľadáme voľnú DE. Ak sa podarilo nájsť voľnú DE (poprípade rozšíriť adresár na ďalší cluster), funkcia vráti spomínaný offset, inak 0. Táto funkcia sa používa pri vytváraní adresárov v EFS a zápise súborov do EFS.

`long GetFreeDEPositions(int DESNeeded,char *path,unsigned long *parentcluster=NULL, unsigned long *NumberOfDEPosition=NULL, unsigned long *NumberOfCluster=NULL,bool *Root=NULL);`	Funkcia podobná predošlej, rozšírená a prispôsobená verzia pre použitie s dlhými názvami. DESNeeded – počet DE, ktoré treba na uloženie dlhého názvu a zodpovedajúceho krátkeho názvu. path – adresár (cesta), v ktorom hľadáme pozíciu pre novú DE (prázdny reťazec pre ROOT). parentcluster – prostredníctvom tohto smerníka (nepovinný parameter) funkcia vráti prvý cluster adresára, v ktorom hľadáme voľnú DE. NumberOfDEPosition – vráti počet už existujúcich DE v clustri – potrebné, ak prvá časť dlhého názvu sa nachádza v jednom clustri adresára a zvyšok v nasledujúcom. NumberOfCluster – vráti číslo clustra, kde sa začína zapisovať dlhý názov, potrebné pri prípadnom prechode na ďalší cluster adresára Root – vráti príznak root-u FAT12/16 – v takomto prípade sa o prechod na ďalší cluster netreba starať Návratová hodnota - vráti offset v súbore EFS, na ktorý možno uložiť prvú novú položku do zadaného adresára, v prípade neúspechu 0.
`unsigned FATEOCMark(void);`	Funkcia vráti minimálnu hodnotu, ktorá sa už považuje za značku konca zreťazenia clustrov (v závislosti od typu FAT).
`int CrtDirectory(char* path, bool longname=false);`	Vytvorí adresár v EFS. Argumentom je názov a cesta vytváraného adresára. Všetky adresáre v ceste musia existovať, posledný bude vytvorený. V prípade úspešného vytvorenia funkcia vráti 1, inak 0. Nepovinný parameter longname slúži na vytvorenie dlhého mena (ak je true).
`int InsertFile(char *RealFileName,char *TargetFileName,bool LongName=false,bool FragmentFile=false);`	Vloží skutočný súbor do EFS. RealFileName – meno skutočného súboru TargetFileName – meno cieľového súboru v EFS LongName – nepovinný parameter, ak je true, funkcia vytvorí pre súbor v EFS dlhý názov FragmentFile – nepovinný parameter, ak je true, funkcia sa pokúsi uložiť súbor do EFS tak, aby bol fragmentovaný (aby nebol uložený v clustroch nasledujúcich za sebou) V prípade úspešného zápisu súboru do EFS funkcia vráti 1, inak 0. **POZOR:** Funkcia netestuje, či sa súbor do EFS zmestí. Ak dôjde miesto uprostred zápisu súboru, funkcia sa ho pokúsi zmazať (netestované).
`bool CreateEmptyFileSystem(char* FileName,int Size, FATType Type,unsigned ClustSize);`	Vytvorí prázdny filesystém s veľkosťou sektora 512 bajtov. FileName – meno súboru pre nový EFS Size – veľkosť EFS v bajtoch. Veľkosť bude v prípade potreby zmenšená na najbližší celý sektor. Možno tiež použiť konštanty HDFLOPPY a DDFLOPPY pre vytvorenie EFS o veľkosti HD/DD diskety Type – typ FAT (FAT12, FAT16, FAT32) ClustSize – veľkosť clustra v sektoroch. Prípustné hodnoty sú 1, 2, 4, 8, 16, 32, 64 a 128. Ak sa filesystém podarilo vytvoriť, funkcia vráti true, inak

	false. Funkcia vracia false aj v prípade zlých vstupných údajov – napr. priveľký počet clustrov v EFS (veľmi veľký EFS a príliš malý cluster). **POZOR: Volanie tejto funkcie je korektné iba pre neinicializovaný objekt triedy FATFileSystem!!!** Pokiaľ sa už predtým volal Init(...), treba vytvoriť nový objekt a tento použiť na vytvorenie prázdneho EFS!!! Takisto možno použiť funkciu *Close(...)*.
`bool SetAttributes(char *NameOfObject, unsigned Attr);`	Nastaví atribúty zadaného objektu EFS(súbor / adresár) NameOfObject – názov (cesta) objektu na zmenu atribútov Attr – atribúty, ktoré budú objektu priradené. Treba ich vytvoriť pomocou logickej funckie OR a konštánt ATTR_READ_ONLY, ATTR_HIDDEN, ATTR_SYSTEM, ATTR_ARCHIVE. Ostatné atribúty sa v tejto funkcii neodporúča používať, mohlo by to viesť k poškodeniu EFS. Užitočná je tiež funkcia *TranslateAttributes()*. Ak sa zmena atribútov úspešne podarila, funkcia vráti true, inak false.
`unsigned TranslateAttributes(char *AttrRet);`	AttrRet – reťazec, ktorý môže obsahovať znaky r, h, s, a, R, H, S, A (v ľub. poradí). Funkcia vráti hodnotu zodpovedajúcu kombinácii atribútov vo vstupnom reťazci, túto hodnotu možno použiť ako argument funkcie *SetAttributes()*. **Príklad:** SetAttributes(„cesta\\súbor“, TranslateAttributes(„rh“)); Toto volanie nastaví súboru atribúty READ ONLY a HIDDEN.
`bool ReadFile(char *EFSName, char *RealFileName);`	Prečíta súbor z EFS do reálneho FS. EFSName – meno súboru v EFS RealFileName – meno cieľového súboru v reálnom FS V prípade úspešného prečítania vráti true, inak false.
`unsigned int GetNumberOfClusters(void);`	Funkcia vráti celkový počet clustrov v EFS (nie najvyššie prípustné číslo clustra).
`DirectoryEntry GetDE(void);`	Vráti aktuálnu hodnotu (nie smerník) dátového člena „de“ objektu FATFileSystem.
`bool WriteDE(char* NameOfObject, DirectoryEntry * inp_de);`	Zapíše Short DirectoryEntry pre zadaný objekt. Použitie: modifikácia dátumu/času, atribútov, apod. NameOfObject – meno objektu v EFS inp_de – smerník na novú DE pre daný objekt, ktorá sa má zapísať. V prípade úspešného zápisu funkcia vráti true, inak false.
`int WriteSomethingToFS(long Offset,void *Ptr, long Lng);`	Táto funkcia zapíše „niečo“ „niekde“ do EFS. Používa sa na výrobu chýb. Offset – pozícia v súbore s EFS Ptr – smerník na dáta, ktoré sa majú zapísať Lng – počet bajtov, ktoré sa majú zapísať Používať uvážlivo!
`void Close(void);`	Ukončí činnosť inštancie triedy FATFileSystem. Po tomto volaní je objekt *neinicializovaný* (možno ho použiť napr. na vytvorenie prázdneho emulovaného súborového systému

	alebo inicializáciu iného súboru s emulovaným súborovým systémom).
`char * GetLastError(void);`	Funkcia vráti smerník na reťazec obsahujúci chybové hlásenie. Volanie tejto funkcie má zatiaľ zmysel iba po **neúspešnom** pokuse o prečítanie súboru z emulovaného súborového systému do reálneho (funkcia *ReadFile(...)* vrátila hodnotu **false**).

Tab. 2.33 – Metódy triedy FATFileSystem

Trieda LongNameCache – dátové členy	
Deklarácia	**Popis**
`wchar_t WideLongName[300];`	Dlhé meno súboru v UNICODE (16-bitové znaky).
`wchar_t LongNameParts[20][13];`	Pole UNICODE znakov pre 13-znakové časti dlhého mena uložené v jednotlivých DE.
`char NarrowLongName[300];`	Dlhé meno súboru v 8-bitových znakoch.
`unsigned char Checksum`	Kontrolný súčet krátkeho mena súboru zodpovedajúceho dlhému.
`unsigned char LastNum`	Index posledne prečítanej časti dlhého mena.
`unsigned char MaxNum`	Počet častí dlhého mena.
`bool Valid`	Dlhé meno je platné.
`bool First`	Bola prečítaná prvá časť dlhého mena.
`bool Last`	Bola prečítaná posledná časť dlhého mena.
`bool Checksummed`	Kontrolný súčet obsiahnutý v častiach dlhých mien sa zhoduje s kontrolným súčtom zodpovedajúceho krátkeho názvu.
`NonPortableLongDirectoryEntry nlde;`	Štruktúra použitá z dôvodu 32-bitového typu **wchar_t** pod OS Linux.

Tab. 2.34 – Dátové členy triedy LongNameCache

Trieda LongNameCache – členské funkcie	
Deklarácia	**Popis**
`void ClearCache(void);`	Inicializuje objekt, pripraví ho na prijímanie častí dlhého názvu alebo jeho vytvorenie
`void ProcessLongEntry(DirectoryEntry de);`	Získa časť dlhého názvu z DE a uloží do internej pamäte. Parametrom je DE. Pri prehľadávaní adresára sa táto funkcia volá po každom prečítaní DE bez ohľadu na typ

	DE(dlhý/krátky). Funkcia zabezpečuje postupné budovanie dlhého názvu z častí, kontrolu častí a porovnanie kontrolných súčtov po prečítaní zodpovedajúceho krátkeho názvu. Ak je príznak *Checksummed* true, potom premenná *NarrowName* obsahuje korektné dlhé meno.
`void ConvertFromUnicode (void);`	konverzia *WideLongName ->> NarrowLongName*. Funkciu netreba volať, volá ju *ProcessLongEntry(...)*.
`void ConvertToUnicode(v oid)`	konverzia *NarrowLongName ->> WideLongName*. Funkciu netreba volať, volá ju *ProcessLongEntry(...)*.
`void ProcessLongName(ch ar *lname);`	Rozdelí dlhé meno lname (**BEZ CESTY!!!**) na časti a zapíše do internej pamäte. Toto volanie predstavuje **1. krok** pri vytváraní nového dlhého mena.
`bool PrepareToWriteLong Name(char *path, char * shortalias, FATFileSystem *ffs);`	Vytvorí krátke meno (SHORT ALIAS) k dlhému menu. Predstavuje **2. krok** vytvorenia nového dlhého mena. path – dlhý názov aj s cestou, kde bude položka s dlhým menom uložená shortalias – na túto adresu funkcia uloží vygenerované krátke meno ffs – smerník na inicializovaný EFS (funkcia ho potrebuje na vygenerovanie jedinečného krátkeho mena) Ak sa vygenerovanie podarilo, funkcia vráti **true**, inak **false**.
`unsigned char ComputeChecksum(ch ar *c);`	Spočíta checksum (návratová hodnota) z 11-znakového reťazca c – mal by to byť krátky názov súboru. Netreba volať, volá ju *ProcessLongEntry(...)*.
`int CreateLDE(Director yEntry *de);`	Táto funkcia naplní DE na adrese de časťou dlhého názvu. Pri prvom volaní vráti poslednú časť dlhého názvu, potom predposlednú, atď. Návratová hodnota funkcie je poradové číslo vrátenej časti dlhého názvu. Treba ju volať dovtedy, kým nevráti 1(prvá časť dlhého názvu) a časti dlhého názvu zapisovať do EFS. **Volanie je korektné len po volaní *PrepareToWriteLongName(...)* a *ProcessLongName(...)*. Táto funkcia predstavuje 3. krok vytvorenia dlhého názvu.**
`void de2nlde(LongDirect oryEntry *src);` `void nlde2lde(LongDirec toryEntry *dst);`	Konverzné funkcie použité z dôvodu 32-bitového typu **wchar_t**, v súborovom systéme FAT sa používajú 16-bitové znaky.

Tab. 2.35 – Metódy triedy LongNameCache

2.3.2. Modul LogSystem (súbory LogSystem.h, LogSystem.cpp)

Modul zabezpečuje záznam činnosti jednotlivých kontrolných programov buď do obyčajného textového súboru alebo do HTML súboru.

Trieda LogSystem – dátové členy	
Deklarácia	**Popis**
`FILE *f;`	Smerník na výstupný súbor.
`LogType type`	Typ logu. Buď textový súbor, alebo HTML.
`bool PREmode;`	Len pre HTML log. Indikuje, že posledný riadok sa nachádza za <PRE> HTML tagom. Pokiaľ sa vypíše do logu ďalší riadok typu *LnTPRE*, netreba tag <PRE> uvádzať, pretože je súčasťou bloku textu so zachovaným pôvodným formátovaním.

Tab. 2.36 – Dátové členy triedy LogSystem

Trieda LogSystem – členské funkcie	
Deklarácia	**Popis**
`bool InitLog(string FileName, LogType Type=LTPlainText, string Title="");`	Inicializuje log. FileName – meno súboru s logom. Ak existuje, bude prepísaný. Type – určuje typ logu – HTML alebo obyčajný text Title – nadpis logu
`void LineToLog(string Msg="", LineType tt=LnTStd, string Color="", string Size="");`	Zapíše riadok do logu. Msg – zapísaný text. tt – typ riadku: - **LnTStd** – štandardný riadok formátovaný browserom - **LnTTT** – riadok formátovaný browserom, bude použitý neproporcionálny font - **LnTPRE** – riadok so zachovaním pôvodného formátovania Color – farba riadku, zadáva sa v HTML tvare RRGGBB, alebo názov Size – veľkosť písma riadku, môže byť relatívna (-2, +1) alebo absolútna Všetky parametre okrem Msg sa týkajú iba HTML logu.
`void CloseLog(void);`	Ukončí činnosť objektu a uzavrie súbor s logom.
`void EFSContentToLog(deque <DirDescr> *EFSContent);`	Funkcia vypíše obsah celého emulovaného FS do logu. Parameter EFSContent sa získa volaním príslušnej funkcie objektu triedy **ExtendedFATFileSystem**.
`LogSystem(void);`	Konštruktor triedy.
`~LogSystem(void);`	Deštruktor triedy.

Tab. 2.37 – Funkcie triedy LogSystem

2.3.3. Modul EFATFS (súbory EFATFS.h, EFATFS.cpp)

Modul obsahuje triedu ExtendedFATFileSystem odvodenú od triedy FATFileSystem. Trieda rozširuje funkcionalitu o metódy používané pri generovaní emulovaných súborových systémov s náhodnou štruktúrou.

Údajové štruktúry

Nasledujúce dve štruktúry sa používajú pri získavaní obsahu existujúceho EFS.

Táto štruktúra obsahuje informácie o jednej položke adresára.

```
struct DirEnt
{
string Name; //obsahuje meno položky, dlhé (ak existuje, inak
krátke)
DirectoryEntry de; //všetky ostatne údaje
};
```

Táto štruktúra obsahuje informácie o jednom adresári EFS.

```
struct DirDescr
{
string DirName;
deque<DirEnt> DirEntries;
};
```

Nasledujúce dve štruktúry sa používajú na popis použitia šablón pri generovaní náhodného EFS.

```
struct DirDesc
     {
     string DirName;
     deque<string> Files_q;
     };
```

Posledná štruktúra popisuje použitie šablón a rozmiestnenie súborov z nich generovaných v EFS.

```
struct FileTemplDesc
     {
     string Template;
     deque<DirDesc> Dirs_q;
     };
```

Trieda ExtendedFATFileSystem – členské funkcie	
Deklarácia	**Popis**
`ExtendedFATFileSystem(void);`	Konštruktor triedy.
`~ExtendedFATFileSystem(void);`	Deštruktor triedy.
`bool GetEFSDir(char *DirName,`	Funkcia uloží obsah adresára DirName (ak existuje v EFS) do existujúceho frontu DirContent (prípadný

`deque<DirEnt> *DirContent, bool DeletedEntries=false);`	predošlý obsah bude vymazaný). Ak je parameter DeletedEntries **true**, potom front bude obsahovať aj zmazané položky.
`void GetAllEFSDirs(deque <DirDescr> *Result);`	Funkcia uloží obsah celého EFS do existujúceho frontu Result. Prípadný predošlý obsah bude zmazaný. Takýto front možno vypísať pomocou triedy **LogSystem**.
`void GenerateFileNameTemplates(deque <string> *Result, string Base, int Num, int ProbM, int ProbA, bool NoMChars=false);`	Funkcia naplní existujúci front Result (prípadný obsah bude zmazaný) Num šablónami mien súborov. Base – základ pre šablónu, znaky # budú v tomto reťazci nahradené: - metaznakom (*, ?) s pravdepodobnosťou ProbM %, pričom * má pravdepodobnosť ProbA % (? 100-ProbA %) - náhodným písmenom alebo číslicou s pravdepodobnosťou (100 - ProbM %) NoMChars - ak je **true**, tolerujú sa šablóny bez metaznakov, inak sú prípustné len šablóny, ktoré obsahujú aspoň jednu hviezdičku alebo aspoň tri otázniky. Nevyžaduje inicializovaný filesystém.
`void GenerateShortFileNameTemplates(deque<string> *Result, int Num);`	Podobná funkcia ako predošlá. Generuje šablóny pre krátke názvy súborov v tvare ########.### a uloží ich do frontu Result. Do šablóny umiestni jeden znak „*“ a prípadne niekoľko ? tak, aby dĺžka mena nepresiahla 11 znakov za predpokladu, že * sa nahradí 0-3 znakmi, zabezpečuje jedinečnosť generovaných šablón. Num – počet generovaných šablón.
`void GenerateFNamesFromTmplts(deque<string> *Result, deque<string> *Templates, int Nmin, int Nmax, int MaxAstCharCnt=10);`	Funkcia naplní existujúci front Result (prípadný obsah bude zmazaný) menami súborov vygenerovanými zo šablón vo fronte Templates (získanom predošlou funkciou): - z každej šablóny sa vygeneruje aspoň Nmin mien, nie však viac ako Nmax mien - znak * v šablóne bude nahradený 0-MaxAstCharCnt náhodnými písmenami alebo číslicami - znak ? v šablóne bude nahradený náhodným písmenom alebo číslicou Mená súborov generované z jednotlivých šablón sú oddelené prázdnym reťazcom. Nevyžaduje inicializovaný filesystém.
`int GenerateRandomFiles (string path, deque<string> *Names, int minsize, int maxsize);`	Funkcia vygeneruje súbory s náhodným obsahom do reálneho FS. Pred každé meno súboru z fronty Names (získanej predošlou funkciou) vloží reťazec (cestu) path. Veľkosť súborov bude aspoň minsize bajtov, neprekročí však maxsize bajtov. Vynechá prázdne reťazce vo fronte mien. Vráti počet vygenerovaných súborov. Nevyžaduje inicializovaný (emulovaný) filesystém.

`void CreateRandomEFSStructure(deque<FileTemplDesc> *Result, deque<string> *Dirs, deque<string> *Files, deque<int> *Attr, deque<string> *Templates, deque<string> *FileNames, int NumOfDirs, int ProbSubDir, int ProbFlInDir, int ProbAR, int ProbAH, int ProbAS, int ProbAA, bool UseShNames=false);`	Táto funkcia vytvorí náhodnú štruktúru emulovaného filesystému (len virtuálne – výsledky uloží do parametrov). Všetky fronty musia existovať. **VSTUPNÉ PARAMETRE:** Templates – front šablón mien súborov získaný funkciou *GenerateFileNameTemplates(...)* FileNames – front mien súborov generovaných zo šablón funkciou *GenerateFNamesFromTmplts (...)* NumOfDirs – počet adresárov, ktoré budú v EFS vytvorené ProbSubDir – pravdepodobnosť (v %) vnorenia vytváraného adresára do už existujúceho adresára ProbFlInDir – pravdepodobnosť (v %) uloženia súboru do náhodne zvoleného adresára ProbAR – pravdepodobnosť (v %) nastavenia atribútu READONLY pre súbor alebo adresár ProbAH – pravdepodobnosť (v %) nastavenia atribútu HIDDEN pre súbor alebo adresár ProbAS – pravdepodobnosť (v %) nastavenia atribútu SYSTEM pre súbor alebo adresár ProbAA – pravdepodobnosť (v %) nastavenia atribútu ARCHIVE pre súbor alebo adresár UseShNames – ak je **true**, funkcia vygeneruje krátke mená adresárov (ich dĺžka nepresiahne 8 znakov) **VÝSTUPNÉ PARAMETRE:** Result – front popisujúci použitie jednotlivých šablón v adresároch, používa sa pri niektorých kontrolách na výber adresárov, ktoré obsahujú určitú šablónu Dirs – front mien vygenerovaných adresárov aj s celou cestou Files – front mien súborov aj s celou cestou (náhodne zaradených do adresárov) Attr – front atribútov, najprv pre adresáre, potom pre súbory Nevyžaduje inicializovaný (emulovaný) filesystém.
`bool FillEFS(string path, deque<string> *Files, deque<string> *FileNames, deque<string> *Dirs, deque<int> *Attr, bool FragmentFiles=false, bool UseShNames=false);`	Funkcia zrealizuje v **inicializovanom** EFS náhodnú štruktúru vygenerovanú predošlou funkciou. path – cesta k súborom (v RFS), ktoré budú vkladané do EFS Files – front mien súborov aj s cestou, získaný predošlou funkciou FileNames – front mien súborov bez ciest získaný funkciou *GenerateFNamesFromTmplts(...)* Dirs – front mien adresárov aj s cestou získaný predošlou funkciou Attr – front atribútov získaný predošlou funkciou FragmentFiles – ak je **true**, systém sa pokúsi zapísať súbory do EFS tak, aby boli fragmentované (výsledok závisí od viacerých faktorov, najmä od objemu zapisovaných údajov – čím menší objem údajov, tým

	väčšia fragmentácia). UseShNames – príznak použitia krátkych mien súborov a adresárov. Ak je **true**, funkcia pri zápise súborov a adresárov do emulovaného súborového systému nevytvorí dlhé názvy.

Tab. 2.38 – Metódy triedy ExtendedFATFileSystem

2.3.4. Modul tokenizer (súbory tokenizer.h, tokenizer.cpp)

Trieda Tokenizer obsiahnutá v tomto module rozdelí vstupný súbor na symboly. Symbol je postupnosť znakov iných ako biele znaky.

Trieda Tokenizer – metódy	
Deklarácia	**Popis**
`bool Init(const char *filename);`	Inicializácia tokenizéra. Vstupom je meno súboru filename. V prípade úspechu vráti **true**, inak **false** (súbor sa nedá otvoriť).
`char * GetBuffAddr(void);`	Vráti adresu buffera obsahujúceho jedno slovo (stačí volať raz, adresa sa nemení).
`int GetNextToken(void);`	Naplní buffer (na adrese vrátenej predošlou funkciou) ďalším symbolom (pokiaľ existuje) a vráti 1. Ak došlo k chybe (preplnenie buffera – symbol dlhší ako 8 KB), vráti -1. Ak v súbore nie je viac symbolov, vráti 0.

Tab. 2.39 – Metódy triedy Tokenizer

2.3.5. Modul ngramanalyzer (súbory ngramanalyzer.h, ngramanalyzer.cpp)

Modul zabezpečuje generovanie n-gramových profilov a ich porovnávanie.

Trieda NgramAnalyzer – metódy	
Deklarácia	**Popis**
`NgramAnalyzer(int TablSze=500009, int Stp=7);`	Konštruktor. Nepovinnými argumentmi sú veľkosť hashovacej tabuľky a krok v prípade kolízie. Musia to byť prvočísla.
`map<string,int>* Analyze(Tokenizer *tok);`	Analýza vstupného textu použitím binárnych usporiadaných stromov. Pomalé. Odporúča sa použiť analýzu pomocou hashovacej funkcie. Argument tok je smerník na inicializovanú inštanciu triedy *Tokenizer*. Návratovou hodnotou je asociatívny kontajner typu *map<string, int>* obsahujúci všetky n-gramy a ich početnosť. Usporiadané podľa n-gramov. V prípade chyby vráti NULL.
`multimap<int, string>* Srt(map<string,int>`	Zotriedi n-gramy vrátené predošlou funkciou podľa početnosti a vráti mapu usporiadanú podľa početnosti.

`* input);`	
`map<string, int> * CreateProfile(multi map <int, string> *input, int maxsize);`	Vytvorí profil. Vstupom je usporiadaná mapa n-gramov input podľa početnosti vrátení predošlou funkciou a veľkosť profilu maxsize (počet n-gramov v profile). Funkcia vráti profil - mapu opäť usporiadanú podľa reťazcov. Táto mapa bude obsahovať najviac maxsize najpočetnejších n-gramov a poradie ich početnosti.
`map<string,int>* AnalyzeWHash(Tokeni zer *tok);`	Analýza vstupného textu použitím hashovacej tabuľky. Argument tok je smerník na inicializovanú inštanciu triedy *Tokenizer*. Návratovou hodnotou je asociatívny kontajner typu *map<string, int>* obsahujúci všetky n-gramy a ich početnosť. Usporiadané podľa n-gramov. V prípade chyby vráti NULL. Typ chyby možno zistiť pomocou funkcie *GetLastError()*.
`char* GetLastError(void);`	Vráti smerník na textový reťazec zodpovedajúci poslednej chybe.
`double ProfileDistance(map <string, int> *prof1, map<string, int> * prof2, double *podobnost=NULL);`	Funkcia zistí vzájomnú vzdialenosť dvoch n-gramových profilov (t. j. podobnosť súborov, z ktorých boli tieto profily vytvorené). Vstupom sú dva profily prof1 a prof2. Návratová hodnota predstavuje vzdialenosť profilov. Pokiaľ sa uvedie nepovinný argument podobnost, funkcia na túto adresu uloží podobnosť profilov v percentách.

Tab. 2.40 – Metódy triedy NgramAnalyzer

2.3.6. Modul xcompat (súbory xcompat.h, xcompat.cpp)

Obsahuje niekoľko pomocných funkcií.

Modul xcompat	
Deklarácia	**Popis**
`void clrscr(void);`	Náhradná funkcia pre prostredie Linuxu, vypíše na konzolu riadok pomlčiek.
`void randomize();`	Inicializuje generátor náhodných čísel pomocou mikrosekúnd aktuálneho času.
`int random(int x);`	Vráti náhodné číslo z intervalu 0, x-1.
`void strupr(char *ret);`	Nahradí malé písmená v reťazci ret veľkými.
`int getchar2(void);`	Vyvolá funkciu *getchar()*, vráti jej hodnotu a potom ju vyvolá ešte raz na elimináciu potvrdzujúceho Enter-u.
`char* fgets_won(char* buf, int maxlen,`	To isté čo *fgets()*, len odstráni znak nového riadku z konca načítaného reťazca.

FILE *stream)	

Tab. 2.41 – Funkcie v module xcompat

2.3.7. Ostatné moduly

Ostatné moduly	
Súbor	**Popis**
efs_content.cpp	Hlavný modul programu *efs_content* (kap. 2.2.1).
efs2rfs_check_f12.cpp	Hlavný modul programu *efs2rfs_check_f12* (kap. 2.2.4).
efs2rfs_check_f16.cpp	Hlavný modul programu *efs2rfs_check_f16* (kap. 2.2.5).
efs2rfs_check_f32.cpp	Hlavný modul programu *efs2rfs_check_f32* (kap. 2.2.5).
efs2rfs_prep_f12.cpp	Hlavný modul programu *efs2rfs_prep_f12* (kap. 2.2.2).
efs2rfs_prep_f16.cpp	Hlavný modul programu *efs2rfs_prep_f16* (kap. 2.2.3).
efs2rfs_prep_f32.cpp	Hlavný modul programu *efs2rfs_prep_f32* (kap. 2.2.3).
ldd_pars.c	Hlavný modul programu *ldd_pars* (kap. 2.2.11).
make_test.c	Hlavný modul programu *make_test* (kap. 2.2.12).
ngramcmp.cpp	Hlavný modul programu *ngramcmp* (kap. 2.2.13).
rfs2efs_check_f12.cpp	Hlavný modul programu *rfs2efs_check_f12* (kap. 2.2.8).
rfs2efs_check_f16.cpp	Hlavný modul programu *rfs2efs_check_f16* (kap. 2.2.9).
rfs2efs_check_f32.cpp	Hlavný modul programu *rfs2efs_check_f32* (kap. 2.2.9).
rfs2efs_prep_f12.cpp	Hlavný modul programu *rfs2efs_prep_f12* (kap. 2.2.6).
rfs2efs_prep_f16.cpp	Hlavný modul programu *rfs2efs_prep_f16* (kap. 2.2.7).
rfs2efs_prep_f32.cpp	Hlavný modul programu *rfs2efs_prep_f32* (kap. 2.2.7).
rfs2efs_prep_f12_shnames.cpp	Hlavný modul programu *rfs2efs_prep_f12_shnames* (kap. 2.2.10).
rfs2efs_prep_f16_shnames.cpp	Hlavný modul programu *rfs2efs_prep_f16_shnames* (kap. 2.2.10).
rfs2efs_prep_f32_shnames.cpp	Hlavný modul programu *rfs2efs_prep_f32_shnames* (kap. 2.2.10).

Tab. 2.42 – Zoznam ostatných modulov zdrojového kódu

2.4. Popis shell skriptov

Skripty vytvárajú prepojenie medzi WWW rozhraním a programami na generovanie a kontrolu emulovaných súborových systémov. Takisto plnia rôzne pomocné úlohy, napr. vytvorenie oddeleného súborového systému pre spustenie zadania, kompiláciu zadania, rozbalenie archívu so zadaním a pod.

2.4.1. efs2rfs_f12.sh

Skript riadi priebeh kontroly zadania čítajúceho súbory z emulovaného súborového systému typu FAT12.

Spustenie a argumenty príkazového riadku:

```
efs2rfs_f12.sh komp zad wdir ddir bdir op
```

Argument	Popis
`efs2rfs_f12.sh`	Názov skriptu.
`komp`	Typ kompresie archívu so zadaním. Prípustné hodnoty sú: • 1 – archív ZIP • 2 – archív TAR • 3 – archív TAR+GZIP
`zad`	Archív so zadaním.
`wdir`	Pracovný adresár, v ktorom bude prebiehať kontrola. Nemusí existovať, skript ho vytvorí. Pokiaľ existuje, jeho obsah bude rekurzívne zmazaný. Názov (cesta) uvedený v tomto argumente **NESMIE** byť zakončený znakom „/“.
`ddir`	Výstupný adresár, do ktorého skript uloží výsledok kontroly. Nemusí existovať, skript ho vytvorí. Pokiaľ existuje, jeho obsah bude rekurzívne zmazaný. Názov (cesta) uvedený v tomto argumente **NESMIE** byť zakončený znakom „/“.
`bdir`	Adresár s programami na prácu s emulovaným súborovým systémom. Skript ich spúšťa počas kontroly. Tento argument **MUSÍ** byť ukončený znakom „/“.
`op`	Ak tento argument obsahuje hodnotu **1**, potom skript iba vytvorí potrebné adresáre a rozbalí archív so zadaním. V opačnom prípade sa vykoná kompilácia zadania, príprava emulovaného súborového systému, spustenie zadania a kontrola jeho činnosti.

Tab. 2.43 – Argumenty príkazového riadku skriptu efs2rfs_f12.sh

Návratové hodnoty:

Hodnota	Význam
0	Vytvorenie potrebných adresárov a rozbalenie archívu so zadaním bolo úspešné (len v prípade použitia argumentu *op* = 1).
1	Rozbalenie archívu so zadaním zlyhalo (len v prípade použitia argumentu *op* = 1). Ak argument príkazového riadku *op* má inú hodnotu ako **1**, skript vždy vráti hodnotu 1.

Tab. 2.44 – Návratové hodnoty skriptu efs2rfs_f12.sh

efs2rfs_f12.sh	
Riadok	**Činnosť**
11 – 30	Funcia *cleanup*. Zabezpečuje korektné ukončenie činnosti skriptu. Výsledky kontroly (súbory *.out, *.err, *.html) skopíruje do výstupného adresára. Rekurzívne vymaže pracovný adresár. Odblokuje semafor (len ak bol zablokovaný) a tým umožní vykonanie kontroly ďalších zadaní. Očakáva 3 argumenty: • pracovný adresár (v ktorom prebiehala kontrola) • výstupný adresár • verdikt – reťazec, ktorý funkcia zapíše do súboru *result.txt* vo výstupnom adresári. Pomocou tohto súboru sa overuje výsledok kontroly.
31 – 97	Táto časť skriptu sa vykoná len v prípade, ak je agrument op rovný **1**. Zvyšok skriptu (po riadku 99) sa vykoná len vtedy, ak argument op nie je rovný **1**.Vykoná nasledujúce činnosti: • zmazanie výstupného a pracovného adresára (ak existovali) • vytvorenie pracovného a výstupného adresára • vytvorenie súboru „.htaccess“ s textom „deny from all“ v obidvoch súboroch – opatrenie zabraňujúce prístup do týchto adresárov cez browser • získanie cesty k súbory s kľúčom semafora riadiaceho front kontroly zadaní, nech je to nadradený adresár pracovného adresára • vytvorenie adresára „nr“ v pracovnom adresári (new root) • skopírovanie archívu so zadaním do tohto adresára • rozbalenie zadania, v prípade chyby skript skončí s návratovou hodnotou 1 • pridelenie práv súborom rozbaleným z archívu • rekurzívny výpis obsahu aktuálneho adresára do dočasného súboru (potrebné pre evidenciu odovzdaných súborov) • ukončenie skriptu s návratovou hodnotou 0 (nadradený PHP skript zaeviduje rozbalené súbory a spustí efs2rfs_f12.sh znovu s parametrom op rovným 0.
107 – 112	Zaradenie sa do frontu zadaní čakajúcich na kontrolu. Skript vyvolá program sb*lock* (kap. 2.2.14), ktorý sa pokúsi znížiť hodnotu semafora riadiaceho kontrolu. Ak sa to podarí (žiadna iná kontrola práve neprebieha), skript pokračuje bez prerušenia ďalej. Ak hodnotu semafora znížiť nemožno, *sembl* zablokuje vykonávanie skriptu (kým neskončí iná kontrola a neuvoľní semafor). Pokiaľ semafor s príslušným kľúčom neexistuje alebo je neprístupný (napr. po reštarte systému), *sembl* vráti 1 a skript sa ukončí s chybou.
118	Zmazanie súboru „*zadanie*“. Mohlo sa stať, že študent už v archíve so zadaním odoslal aj skompilovaný binárny súbor, tento zmažeme, pretože súčasťou zadania je aj preverenie úspešnosti kompilácie zadania.
120 –	Súčasťou zadania musí byť aj súbor *makefile*, ktorý zabezpečí kompiláciu

127	zadania pomocou utility *make* a vznik spustiteľného súboru „zadanie“. Tieto riadky skriptu kontrolujú, či *makefile* neobsahuje nejaké iné príkazy okrem volania kompilátora (kompilácia neprebieha v oddelenom súborovom systéme a *makefile* umožňuje vykonať akékoľvek príkazy). Využíva program *make_test* (kap. 2.2.12). V prípade zistenia nepovolených príkazov sa skript ukončí.
130 – 135	Kompilácia pomocou utility *make*. Ak došlo k chybe, skript sa ukončí.
139 – 142	Príprava pracovného emulovaného súborového systému. Pozostáva z vytvorenia konfiguračného súboru *time_cfg* a spustenia programu na prípravu emulovaného súborového systému typu FAT12 *efs2rfs_prep_f12* (kap. 2.2.2). Obsah spomínaného konfiguračného súboru zabezpečí: • generovanie skriptu na spustenie zadania obsahujúceho príkaz *chroot* (oddelený súborový systém) • obmedzenie maximálneho času behu zadania na 60 sekúnd (celkový čas) • obmedzenie dostupného času procesora na 50 sekúnd (cez dodatočné parametre programu *runner*, kap. 2.2.15) • obmedzenie veľkosti virtuálneho pamäťového priestoru zadania na cca 64 MB • obmedzenie maximálnej veľkosti súboru vytvoreného zadaním na cca 50 MB
144 – 149	Záloha pracovného emul. FS a presunutie zoznamov súborov (súbory _to_check a _to_check2) na skopírovanie do nadradeného adresára (aktuálny adresár sa stane novým koreňovým adresárom, tieto súbory nebudú dostupné pre zadanie). Keďže tento typ zadania nemá zapisovať do emulovaného súborového systému, jeho zálohou a potom obnovením pred kontrolou sa vyhneme problémom v prípade poškodenia emul. FS zadaním.
151 – 157	Príprava nového koreňového adresára. Vytvorenie podadresárov bin, lib a dev.
157 – 161	Prekopírovanie potrebných programov do oddeleného súborového systému (do adresára bin, ide o programy *runner, chmod* a *du*.)
163 – 174	Zistenie potrebných dynamických knižníc pre tieto programy a zadanie. Využitie programu *ldd_pars* (kap. 2.2.11) na vytvorenie skriptu kopírujúceho tieto knižnice do adresára lib v budúcom oddelenom súborovom systéme. Spustenie a následné vymazanie tohto skriptu a ďalších pomocných súborov.
176 – 178	Spustenie zadania – zabezpečí to skript *_to_run* (kap. 2.4.3) generovaný programom na prípravu emul. FS *efs2rfs_prep_f12* (kap. 2.2.2).
183	Keďže zadanie bežalo v oddelenom súborovom systéme (ale pod root-om, root-ovské práva sú potrebné na vytvorenie oddeleného filesystému), treba zmeniť práva pre všetky súbory vytvorené zadaním (vyvolaním príkazu *chmod* v oddelenom filesystéme).
185 – 190	Nenulová návratová hodnota skriptu *_to_run* signalizuje možnosť preplnenia disku – zadanie vytvorilo väčší výstup ako 10 MB. V takomto prípade sa kontrola predčasne ukončí.
192 – 213	Po úspešnom spustení zadania nasleduje spustenie kontrolného programu (*efs2rfs_check_f12*, skontroluje činnoť zadania, kap. 2.2.4). Použijeme aj program *runner* pre prípad zacyklenia alebo pádu kontrolného programu. Na základe jeho návratovej hodnoty určíme výsledok kontroly. Spusteniu

	kontrolného programu predchádza obnova EFS zo zálohy a presunutie zoznamov súborov do aktuálneho adresára (pozri riadky 144 – 149).

Tab. 2.45 – Detailný popis činnosti skriptu efs2rfs_f12.sh

2.4.2. efs2rfs_f16.sh, efs2rfs_f32.sh, rfs2efs_f12.sh, rfs2efs_f16.sh, rfs2efs_f32.sh, rfs2efs_f12_shnames.sh, rfs2efs_f16_shnames.sh, rfs2efs_f32_shnames.sh

Analogické skripty líšiace sa iba spúšťanými programami (pre ostatné typy zadaní).

2.4.3. _to_run

Skript generovaný programom na prípravu emulovaného súborového systému (*efs2rfs_prep_f12*, *efs2rfs_prep_f16*, *efs2rfs_prep_f32*, *rfs2efs_prep_f12*, *rfs2efs_prep_f16*, *rfs2efs_prep_f32*, *rfs2efs_prep_f12_shnames*, *rfs2efs_prep_f16_shnames*, *rfs2efs_prep_f32_shnames*, kap. 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.6, 2.2.7, 2.2.8, 2.2.9, 2.2.10). Zabezpečuje spustenie zadania s príslušnými parametrami a obmedzenie maximálneho času behu zadania pomocou programu *runner* (kap. 2.2.15). Takisto po každom spustení zadania overuje, či nevytvorilo výstup väčší ako 10 MB. Ak áno, skript sa ukončí s návratovou hodnotou 1 (inak vráti 0). Ďalej môže spúšťať zadanie v oddelenom súborovom systéme (sudo + chroot) a obmedziť dostupné systémové prostriedky pre zadanie (podľa konfiguračného súboru *time_cfg*).

2.4.4. DelDir.sh

Skript očakáva jeden argument príkazového riadku – adresár. Po piatich minútach od spustenia tohto skriptu dôjde k rekurzívnemu zmazaniu adresára uvedeného ako argument príkazového riadku. Používa sa na zmazanie dočasných adresárov pri downloade zadaní učiteľom.

2.4.5. similarity_automatic.php

Neinteraktívny PHP skript určený na spustenie z príkazového riadku. Vykoná výpočet podobnosti odovzdaných zadaní. Je vhodný na pravidelné spúšťanie napr. cez *cron*.

2.5. PHP skripty

Tvoria www rozhranie systému. Skripty sú rozdelené na tri skupiny (podľa používateľa):

- skripty pre administrátora (môže ich spúšťať iba administrátor): ich názov sa začína reťazcom „admin“
- skripty pre učiteľa (môže ich spúšťať iba učiteľ): ich názov sa začína reťazcom „teacher“
- skripty pre študenta (môže ich spúšťať iba študent): ich názov sa začína reťazcom „student“

2.5.1. admin.php

Hlavné menu administrátora systému. Obsahuje nasledujúce možnosti:

- správa učiteľov

- správa zadaní
- zmena osobných údajov (meno a priezvisko administrátora)
- vytvorenie tabuliek pre aktuálny akad. rok
- import študentov z LDAP servera
- zobrazenie logov
- zmena textov na úvodnej stránke systému
- analýza podobnosti odovzdaných súborov
- aktualizácia databázy odovzdaných súborov
- funkcia „su“ – zmena identity administrátora na ktoréhokoľvek učiteľa alebo študenta v systéme
- zmena vlastného hesla
- koniec, odhlásenie sa zo systému

2.5.2. admin_fa_rebuild.php

Skript zabezpečujúci aktualizáciu databázy odovzdaných súborov. Táto databáza sa buduje priebežne po uploade každého archívu so zadaním. Niekedy sa však môže stať, že obsahuje neplatné údaje (napr. reštart servera tesne po uploade zadania, ale pred uložením odovzdaných súborov do databázy). Jedným z prejavov nekorektných údajov v databáze odovzdaných súborov môžu byť chyby rozbalenia súborov pri výpočte podobnosti odovzdaných zadaní. Databáza neobsahuje súbory typu *makefile*.
Skript vykoná nasledujúce akcie:

- uzamkne tabuľku odovzdaných súborov, podobnosti súborov a spolupracujúcich študentov
- vymaže aktuálny obsah týchto tabuliek
- postupne rozbalí všetky odovzdané archívy a ich obsah uloží do tabuľky odovzdaných súborov
- skript spracuje len archívy pre aktuálny akad. rok (akad. rok, meno, priezvisko a id študenta, ktoré sú súčasťou názvu archívu, sa musia zhodovať s tabuľkou študentov aktuálneho akad. roku)

Spustením tohto skriptu sa vymažú všetky informácie o podobnosti odovzdaných súborov. Po tejto operácii treba znovu vypočítať podobnosť súborov a zistiť spolupracujúcich študentov (kap 2.5.7, 2.4.5).

2.5.3. admin_change.php

Jednoduchý skript zabezpečujúci zmenu mena a priezviska administrátora systému (povolené sú iba písmená a číslice).

2.5.4. admin_ldap_import.php

Skript zabezpečujúci import študentov pre aktuálny akad. rok z LDAP servera do tabuľky študentov. Využíva konštanty definované v súbore **ldap_params.php** (kap. 2.5.23). Postupuje takto:

- pokus o pripojenie k definovanému LDAP serveru (konštantou **LDAP_SERVER**)
- nastavenie protokolu na verziu 3
- anonymný bind
- vyhľadanie študentov so zapísaným predmetom operačné systémy:

- ako základ vyhľadania sa použije konštanta („LDAP Search Base“) **LDAP_SEARCH_BASE** (v súčasnosti reťazec „ou=People,dc=fei,dc=tuke,dc=sk“)
- ako filter sa použije konštanta **LDAP_SEARCH_FILTER** (v súčasnosti reťazec „accessTo=OS“)
- skript požiada LDAP server o nasledujúce atribúty o vyhľadaných študentoch:
    - meno (konštanta **LDAP_ATTR_NAME**, aktuálne „cn“, predpokladá sa, že tento atribút obsahuje meno a priezvisko oddelené medzerou)
    - identifikačné číslo študenta (konštanta **LDAP_ATTR_ID**, aktuálne „employeenumber“)
    - login študenta (konštanta **LDAP_ATTR_LOGIN**, aktuálne „userid“)
- vyhľadaní študenti budú zaradení do tabuľky študentov (zaradenie zlyhá, ak v tabuľke už existuje študent s rovnakým loginom ako v LDAP-e, loginy musia byť jedinečné, preto sa odporúča najprv urobiť import a až potom doplniť ostatných študentov)

2.5.5. admin_logs.php

Skript umožňuje administrátorovi systému zobraziť log súbory. V skutočnosti ide o tabuľky v databáze, ktorých názov sa začína predponou „**log_**“. Aktuálne sa používajú **log_login_log** (záznamy o prihlásení a odhlásení používateľov) a **log_misc_log** (rôzne významnejšie udalosti a chyby v systéme).

2.5.6. admin_msgs.php

Skript slúži na vytváranie, zmenu a mazanie textových správ zobrazených na prihlasovacej stránke systému. Sú použiteľné na oznamy, upozornenia, novinky, a pod.

2.5.7. admin_similarity2.php

Výpočet podobnosti odovzdaných zadaní. Predpoklady pre výpočet:

- tabuľka odovzdaných súborov (názov, veľkosť, vlastník, ...)
- hash (MD5) každého odovzdaného súboru
- tabuľka podobnosti už porovnaných súborov

Výpočet prebieha v týchto krokoch:

- uzamknutie príslušných tabuliek
- zistenie **identických** súborov odovzdaných viacerými študentmi (systém považuje 2 súbory za identické práve vtedy, ak ich hash a veľkosť sú rovnaké)
- zistenie **podobných** súborov:
    - súbory s podobným obsahom budú mať podobnú veľkosť, teda najprv nájdeme dvojice súborov patriace rôznym študentom s odchýlkou veľkosti maximálne 5%
    - zistíme podobnosť týchto dvojíc pomocou n-gramov (pomocou externého programu *ngramcmp*) a uložíme ju do tabuľky podobnosti
    - podobnosť dvojice sa určuje len v prípade, že nie je v tabuľke podobnosti

- na základe identických a podobných súborov sa určia dvojice spolupracujúcich študentov
- výpočet môže trvať dosť dlho
- tento skript slúži na interaktívny výpočet podobnosti (možno nastaviť maximálnu odchýlku veľkosti porovnávaných súborov a hranicu pre určenie spolupráce, tieto údaje sa však neodporúča meniť – iné skripty ich zmenu nezohľadňujú, tiež možno vypísať porovnávané súbory)
- na automatizovaný pravidelný výpočet podobnosti slúži skript **similarity_automatic.php** v adresári **storage/bin** (kap. 2.4.5)

2.5.8. admin_su.php

Zmena identity administrátora systému. Stačí zvoliť ľubovoľného používateľa systému (študenta alebo učiteľa) a skript vykoná prihlásenie v jeho mene.

2.5.9. admin_teachers.php

Skript rieši správu učiteľov administrátorom:

- pridanie nového učiteľa
- zrušenie existujúceho učiteľa
- zmena osobných údajov (meno, priezvisko, login) učiteľa
- generovanie nového hesla pre učiteľa (len pre internú autentifikáciu)
- zmena autentifikácie učiteľa (z internej na LDAP a opačne)

Pri zavedení nového učiteľa autentifikovaného cez LDAP alebo pri zmene autentifikácie existujúceho učiteľa na LDAP sa systém pokúsi zistiť jeho login (anonymným pripojením na LDAP server, pomocou zadaného mena a priezviska, analogicky ako pri importe študentov, kap. 2.5.4). Pri zavedení nového učiteľa autentifikovaného interne alebo pri zmene autentifikácie existujúceho učiteľa na internú sa systém pokúsi vygenerovať login a heslo. Generovanie loginu spočíva v pripojení prvého písmena mena za priezvisko, prevod na malé písmená a odstránenie diakritiky. Pokiaľ takýto login už existuje, potom sa k nemu ešte pripojí číslo podľa potreby (napr. Ján Novák bude mať login novakj, potom Jozef Novák bude mať login novakj1).

2.5.10. admin_year.php

Skript na vytvorenie tabuliek pre nový akademický rok v databáze. Skript treba spustiť na začiatku akad. roku (t. j. v septembri). Tabuľky platiace len pre jeden akad. rok obsahujú tento rok v svojom názve, v súčasnosti sú to tieto:

- student_xxxy_xxxz – tabuľka študentov akad. roku xxxy/xxxz
- files_xxxy_xxxz – tabuľka odovzdaných súborov v akad. roku xxxy/xxxz
- files_similarity_xxxy_xxxz – tabuľka podobnosti súborov v akad. roku xxxy/xxxz
- spolupraca_xxxy_xxxz – tabuľka spolupracujúcich študentov pre akad. rok xxxy/xxxz

Bližší popis tabuliek je v časti 2.1 Popis databázy. Skript okrem vytvorenia týchto tabuliek vymaže logy a tabuľku študijných skupín. Definície (MySQL) týchto tabuliek sú v súbore **student_table.php** (kap. 2.5.32).

2.5.11. admin_zad.php

Umožňuje správu zadaní:

- pridať / zrušiť / editovať zadanie
- priradiť shell skript zadaniu, ktorý vykoná príslušnú kontrolu

Znenie zadania môže byť vo formáte HTML, max. 64KB.

2.5.12. conn_params.php

Dôležitý skript. Obsahuje parametre pripojenia k databáze. Definuje nasledujúce konštanty:

- **DB_SERVER** – databázový (MySQL) server (prázdny reťazec pre localhost)
- **DB_USER** – meno používateľa, ktoré bude systém používať na prihlásenie sa k databáze
- **DB_NAME** – názov databázy
- **DB_PASSWORD** – heslo (pre pripojenie bez hesla uvedieme prázdny reťazec)

2.5.13. functions.php

Rôzne pomocné funkcie.

- *RemoveDiakr($ret)* – v reťazci ret nahradí diakritické znaky normálnymi
- *SkRok()* – vráti aktuálny školský rok ako reťazec v tvare xxxz/xxxy, napr. 2003/2004
- *SkRok2()* – to isté, ale roky sú spojené podčiarkovníkom
- *TestPrivileges($priv)* – funkcia overí, či daný používateľ má právo spustiť určitý skript (porovná typ používateľa s parametrom priv). Ak áno, vráti **true**, inak vráti **false**, zruší aktuálnu session a používateľa presmeruje na stránku **invalid.php**. Parameter priv je reťazec obsahujúci práva potrebné na spustenie daného skriptu. V systéme sú zavedené 3 úrovne prístupu:
  - admin
  - ucitel
  - student
- *testlegalret($ret*) – vráti **true**, ak reťazec pozostáva len z písmen a číslic. Ak obsahuje nejaké iné znaky, vráti **false.**
- *testlegalretd($ret*) – to isté, čo predošlá funkcia, ale navyše toleruje diakritiku
- *GenerateLinkButton($link,$popis)* – vygeneruje HTML formulár, ktorý obsahuje jedno tlačidlo typu „submit“ s textom popis, ktoré po kliknutí presmeruje browser na adresu link
- *GenerateLinkButtonWG($link,$popis)* – predchádzajúca funkcia rozšírená o uchovanie zvolenej skupiny (študentov) a triedenia (do generovaného formulára pridá ako skryté vstupné prvky obsah premenných *$_SESSION[“skup”]* a *$_SESSION[“srt“]*.
- *make_seed()* – funkcia inicializuje generátor náhodných čísel na základe aktuálneho času
- *GenerString($x)* – vygeneruje náhodný reťazec dĺžky x znakov. Reťazec obsahuje len malé a veľké písmená a číslice
- *tabletoy($table)* – zmení reťazec table tvaru student_xxxx_xxxy na xxxx/xxxy. Zmení názov tabuľky na evidenciu študentov pre aktuálny akad. rok na rok.
- *ytotable($y)* – inverzná funkcia k predošlej
- *CurYearTest()* – funkcia otestuje, či prihlásený učiteľ **má** zvolený aktuálny školský rok (vráti *true*, inak *false*)
- *NotCurYearTest()* - funkcia otestuje, či prihlásený učiteľ **nemá** zvolený aktuálny školský rok (vráti *true*, inak *false*)

- *GeneratePath()* – generuje jedinečnú cestu pre každého študenta v tvare xxxx_xxxy_priezvisko_meno_id_ pre aktuálny šk. rok, z mena a priezviska odstráni prípadnú diakritiku
- *GeneratePath2(), GeneratePath3()* – výsledok je rovnaký ako v predošlom prípade, ale funkcia sa používa z pohľadu učiteľa a zohľadňuje zvolený šk. rok (príslušné údaje sú uložené v iných premenných)
- *GetLDAPLoginByNameAndID($name,$id)* - získa login z LDAP servera pre používateľa na základe mena a identifikačného čísla (pre študenta). Možné návratové hodnoty:
    - *0* – chyba komunikácie s LDAP serverom
    - *-1* – študent nie je v LDAP-e alebo je nájdených osôb viac
    - v prípade úspešného vykonania funkcia vráti požadovaný login

Funkcia využíva anonymný prístup k LDAP serveru a konštanty definované v súbore **ldap_params.php** (kap. 2.5.23) podobne ako skript **admin_ldap_import.php** (kap. 2.5.4).

- *GetLDAPLoginByName($name)* – analogická funkcia, vstupným argumentom je však len meno osoby (učiteľa)
- *LDAPAuthUser($login,$pwd)* – autentifikácia používateľa prostredníctvom LDAP servera. Argumentmi sú login (login) a heslo (pwd). Funkcia vráti **true** v prípade úspešnej autentifikácie, inak **false**. Funkcia využíva konštanty definované v súbore **ldap_params.php** (kap. 2.5.23) podobne ako skript **admin_ldap_import.php** (kap. 2.5.4).
- *CountDirsInDir($i_dir)* – funkcia vráti počet podadresárov v adresári i_dir. Neráta „.“ a „..“.
- *ExtractFile($archive, $file, $destpath, $destname)* – rozbalí požadovaný súbor z archívu. Argumenty:
    - archive – **relatívna** cesta k archívu
    - file – súbor, ktorý má byť rozbalený
    - destpath – cieľový adresár
    - destname – cieľové meno – rozbalený súbor funkcia premenuje (prípadne presunie do cieľového adresára, pretože v archíve mohol byť v nejakom podadresári) na destname

Ďalej funkcia zmaže prípadné podadresáre vzniknuté počas rozbalenia v cieľovom adresári. V prípade úspechu vráti **true**, inak **false**. Používa sa pri výpočte podobnosti súborov.

- *FileSimilarity($file1, $file2)* – zistí podobnosť súborov file1 a file2 pomocou programu *ngramcmp*. Vráti podobnosť v percentách alebo **false** v prípade chyby. Predpokladá, že do adresára (adresárov), kde sú súbory file1 a file2 možno zapisovať.
- *function FileSimilarity2($tdir, $files, &$errors)* – zistí podobnosť viacerých súborov. Argumenty:
    - tdir – pracovný adresár
    - files – pole obsahujúce mená súborov na porovnanie. Každý prvok poľa je reťazec obsahujúci mená porovnávaných súborov oddelené medzerou, pred každé meno bude pripojená cesta tdir.
    - errors – výstupný argument, počet dvojíc, kde pri porovnávaní došlo k chybe

Návratovou hodnotou funkcie je pole podobností. Každý prvok je podobnosť dvojice súborov v percentách. Prvky tohto poľa zodpovedajú prvkom vstupného poľa files.

- *getmicrotime()* – vráti aktuálnu časovú známku (timestamp) aj s milisekundami ako číslo v pohyblivej rádovej čiarke
- *DisplayableFile($fname)* – funkcia zisťuje, či ide o zobraziteľný (nebinárny) súbor (na základe prípony súboru). Ak áno, vráti **true**, inak **false**.
- *ExtractAllFiles($archive, $destpath)* – funkcia rozbalí archív archive do adresára destpath. Podľa výsledku operácie vráti **true** alebo **false**.
- *GetOrigFiles($link,$id)* – funkcia vráti podiel originálnych súborov pre študenta identifikovaného číslom id (v percentách). Využije pripojenie na databázu link.
- *PrepTempStudTbl($link)* – vytvorí dočasnú tabuľku „tab“ študentov usporiadanú podľa dátumu odovzdania, obohatenú o stĺpec udávajúci poradie odovzdania. Využije pripojenie na databázu link.
- *GetCommonFiles($link,$id1, $id2)* – vráti podiel spoločných súborov študentov identifikovaných v tabuľke študentov číslami id1 a id2 (v percentách). Hodnota udáva, akú časť zo súborov odovzdaných prvým študentom odovzdal aj druhý študent (funkcia uvažuje zhodné aj dostatočne podobné súbory – podobnosť aspoň 90%). Využije pripojenie na databázu link.
- *explain_file($f)* – funkcia slúži na vysvetlenie významu súborov vytvorených počas kontroly zadania. Argument f je názov takéhoto súboru, funkcia vráti reťazec popisujúci funkciu tohto súboru. Pokiaľ nie je popis definovaný, funkcia vráti prázdny reťazec.

2.5.14. header.php

Hlavička stránky pri prihlasovaní sa do systému.

2.5.15. header_a.php

Hlavička stránky pre administrátora. Zobrazí ju každý skript pre administrátora.

2.5.16. header_s.php

Hlavička stránky pre študenta. Zobrazí ju každý skript pre študenta.

2.5.17. header_t.php

Hlavička stránky pre učiteľa. Zobrazí ju každý skript pre učiteľa.

2.5.18. change_pwd.php

Zabezpečuje zmenu hesla pre ľubovoľného riadne prihláseného používateľa.

2.5.19. i2.php

Prihlasovací formulár. Pod ním sú zobrazené textové správy (administrátor ich môže meniť, použiteľné na oznamy, upozornenia, novinky, a pod.).

2.5.20. index.php

Úvodná stránka systému.

2.5.21. invalid.php

Stránka obsahujúca oznam o nepovolenom prístupe do systému. Browser je na ňu presmerovaný v prípade neúspešného prihlásenia alebo pri pokuse používateľa spustiť skript nezodpovedajúci jeho oprávneniam.

2.5.22. invalid2.php

Stránka podobná predošlej, ale naviac obsahuje oznam o použití nedovolených znakov v logine alebo hesle (povolené sú len malé a veľké písmená a číslice).

2.5.23. ldap_params.php

Skript definuje konštanty potrebné na komunikáciu s LDAP serverom (import študentov, autentifikácia používateľov systému, pozri tiež 2.5.4, 2.5.9, 2.5.13).

2.5.24. login.php

Skript zabezpečuje prihlásenie používateľa. Najprv overí, či sa v logine nenachádzajú nepovolené znaky (pozri **functions.php**, kap. 2.5.13, funkcia *testlegalret(...)*). Potom sa pripojí na databázu a overí login najprv v tabuľke pre učiteľov a potom aj v tabuľke pre študentov aktuálneho šk. roku. Ak sa login v žiadnej z uvedených tabuliek nenachádza, presmeruje browser na stránku **invalid.php**. Ak má daný používateľ nastavenú LDAP autentifikáciu, odošle login a heslo LDAP serveru. V opačnom prípade overí heslo v príslušnej tabuľke databázy. Urobí záznam o prihlásení do logu login_log. Ak bolo prihlásenie úspešné, potom používateľa presmeruje na príslušnú stránku podľa jeho práv (**student.php**, **teacher.php**, **admin.php**). Po prihlásení študenta skript navyše overí funkčnosť semafora riadiaceho front kontroly zadaní a v prípade potreby vytvorí nový semafor (pozri 2.4.1 a 2.2.14).

2.5.25. logout.php

Zabezpečí odhlásenie používateľa. Urobí záznam o odhlásení do logu login_log. Zruší platnosť aktuálnej session. Ak používateľ naplnil dočasné tabuľky (temptab_stud, temptab_uc), vymaže ich obsah.

2.5.26. mysql_univ_db.php

Modul zabezpečujúci abstrakciu databázových funkcií pre databázu MySQL.

2.5.27. path_cfg.php

Modul definuje konštanty nastavujúce relatívne cesty k rôznym súčastiam systému:

- uchovávanie zadaní: **ZADPATH**
- pracovný adresár pre kontrolu zadaní: **CHECKPATH**
- adresár pre uchovanie výsledkov kontroly zadaní: **RESULTPATH**
- adresár obsahujúci spustiteľné programy a skripty: **SCRIPTPATH**
- pracovný adresár na sťahovanie zadaní: **DOWNPATH**
- dočasný adresár: **TEMPPATH**

Všetky cesty musia byť ukončené znakom „/“.

2.5.28. student.php

Hlavné menu pre študenta. Obsahuje možnosti:

- prehľad znenia všetkých zadaní
- vlastné zadanie (odovzdanie, výsledky kontroly, znenie ...)
- zmena hesla
- koniec (odhlásenie zo systému)

2.5.29. student_odovzd.php

Skript najprv overí, či študent môže odovzdať zadanie:

- študent má pridelené zadanie
- neuplynul termín odovzdania
- ani jedna predošlá kontrola zadania nebola úspešná na 100%
- predošlá kontrola zadania sa už skončila a študent si pozrel jej výsledok

Ak sú všetky uvedené podmienky splnené, skript poskytne inštrukcie na odovzdanie (upload) zadania a vygeneruje príslušný HTML formulár.

2.5.30. student_results.php

Sprístupní študentovi výsledky kontroly naposledy odovzdaného zadania a vyhodnotí ich (ak je to možné – kontrola neprebieha a zadanie bolo už aspoň raz odovzdané).

2.5.31. student_result_all.php

Sprístupní študentovi výsledky kontroly všetkých odovzdaných zadaní (okrem posledného odovzdania).

2.5.32. student_table.php

Skript obsahuje (MySQL) definíciu databázových tabuliek, ktoré treba vytvoriť na začiatku každého akad. roku (pozri tiež kap. 2.5.10).

2.5.33. student_upload.php

Skript zabezpečuje spracovanie archívu so zadaním odoslaného na server. Ak ide o súbor správneho typu a veľkosti (**.zip**, **.tar**, **.tar.gzip**, max. 100 KB), vykoná nasledujúce činnosti:

- uloží zadanie do archívu (do adresára daného konštantou **ZADPATH**)
- vytvorí dočasný podadresár v adresári na kontrolu zadaní (**CHECKPATH**)
- spustí príslušný skript zabezpečujúci kontrolu, tento iba rozbalí archív so zadaním
- uloží do databázy odovzdané súbory
- ak bolo rozbalenie archívu so zadaním úspešné, opäť sa spustí shell skript zabezpečujúci kontrolu, teraz už prebehne kontrola zadania

Pozri tiež Popis shell skriptov, efs2rfs_f12.sh (kap. 2.4.1) a PHP skripty, path_cfg.php (kap. 2.5.27).

2.5.34. student_vsetky_zadania.php

Zobrazí študentovi prehľad tém všetkých zadaní s možnosťou zobrazenia ich plného znenia.

2.5.35. student_zadanie.php

Stránka s informáciami o pridelenom zadaní. Obsahuje stav zadania, možnosti odovzdania, zobrazenia výsledkov kontroly a plného znenia zadania.

2.5.36. student_znenie1zad.php

Zobrazí študentovi plné znenie konkrétneho zadania (buď vlastného alebo vybratého z ponuky všetkých zadaní).

2.5.37. teach_students.php

Správa študentov učiteľom pre aktuálny šk. rok. Analogické ako admin_teachers.php s tým rozdielom, že učiteľ vidí len „svojich“ študentov (tých, ktorých učí).

2.5.38. teacher.php

Hlavné menu pre učiteľa. Obsahuje:

- zobrazenie znenia všetkých zadaní v systéme
- správa študentov aktuálneho akad. roku (pridať, odstrániť, meno, login, heslo, ...)
- správa skupín študentov pre aktuálny akad. rok
- správa zadaní študentov aktuálneho akad. roku
- hromadné pridelenie zadaní
- archív zadaní z minulých rokov
- zobrazenie logov
- zmena aktuálneho akad. roku
- zmena hesla
- koniec (odhlásenie zo systému)

2.5.39. teacher_archiv.php

Archív zadaní pre niektorý z uplynulých rokov (na základe zvoleného roku). Umožňuje:

- zobrazenie všetkých študentov, tém ich zadaní, stavu ich zadaní (nie len tých, ktorých daný učiteľ učil)
- download zadania ľubovoľného študenta
- zobrazenie poznámok k hodnoteniu daného študenta
- zobrazenie výsledkov kontroly zadaní

2.5.40. teacher_down.php

Umožňuje učiteľovi stiahnuť súbor (zadanie). Keďže do adresára so zadaniami www server nemá prístup, v adresári DOWNPATH sa vytvorí jedinečný adresár s povoleným prístupom. Do tohto adresára sa skopíruje súbor so zadaním. Na pozadí sa spustí skript, ktorý zabezpečí zmazanie tohto adresára o 5 minút. Nakoniec bude browser presmerovaný na tento súbor.

2.5.41. teacher_down_list.php

Skript generuje pre učiteľa zoznam stiahnuteľných súborov odovzdaných konkrétnym študentom.

2.5.42. teacher_groups.php

Pre učiteľa zabezpečuje správu skupín študentov (vytvorenie skupiny, zaradenie študenta do skupiny, preradenie študenta medzi skupinami, zmena učiteľa skupiny, zrušenie skupiny).

2.5.43. teacher_hromadzad.php

Skript umožňuje učiteľovi tzv. hromadné pridelenie zadaní, t. j. vygeneruje formulár, kde možno prideliť zadania študentom jednej študijnej skupiny súčasne.

2.5.44. teacher_logs.php

Skript umožňuje učiteľovi zobraziť log súbory. V skutočnosti ide o tabuľky v databáze, ktorých názov začína predponou „**log_“**. Aktuálne sa používajú **log_login_log** (záznamy o prihlásení a odhlásení používateľov) a **log_misc_log** (rôzne významnejšie udalosti a chyby v systéme).

2.5.45. teacher_results.php

Sprístupňuje učiteľovi výsledky kontroly zadania ľubovoľného študenta (buď z archívu, alebo z aktuálneho roku).

2.5.46. teacher_spolupr.php

Zobrazuje učiteľovi detailné informácie o potenciálnej spolupráci študentov na základe výpočtu podobnosti odovzdaných zadaní.

2.5.47. teacher_vsetky_zadania.php

Skript poskytuje učiteľovi zoznam všetkých dostupných zadaní v systéme s možnosťou zobrazenia ich plného znenia.

2.5.48. teacher_year.php

Zabezpečuje voľbu akad. roku pre učiteľa.

2.5.49. teacher_zadrefs.php

Správa zadaní učiteľových študentov pre aktuálny rok.

- zobrazuje študentov, tému ich zadania, stav zadania
- umožňuje prideliť/zmeniť zadanie a dátum odovzdania
- ak zadanie bolo aspoň raz odovzdané, možno zobraziť výsledky kontroly a stiahnuť ho
- možno napísať/zmeniť poznámky k hodnoteniu študenta (max. 64KB)

2.5.50. teacher_znenie1zad.php

Zobrazí učiteľovi plné znenie konkrétneho zadania.

2.5.51. univ_db.php

Modul abstrakcie databázových funkcií PHP. Pôvodne mal tento skript spolu so skriptom **mysql_univ_db.php** umožniť jednoduché prispôsobenie systému na iný databázový server (nahradením skriptu **mysql_univ_db.php**). V skutočnosti by zmena databázového servera nebola až taká jednoduchá, pretože systém používa množstvo špecifických MySQL dotazov.

3. Preklad C/C++ programov

3.1. Zoznam zdrojových textov

EFATFS.cpp
EFATFS.h
efs2rfs_check_f12.cpp
efs2rfs_check_f16.cpp
efs2rfs_check_f32.cpp
efs2rfs_prep_f12.cpp
efs2rfs_prep_f16.cpp
efs2rfs_prep_f32.cpp
efs_content.cpp
fatlib.cpp
fatlib.h
ldd_pars.c
LogSystem.cpp
LogSystem.h
Makefile
make_test.c
ngramanalyzer.cpp
ngramanalyzer.h
ngramcmp.cpp
rfs2efs_check_f12.cpp
rfs2efs_check_f16.cpp
rfs2efs_check_f32.cpp
rfs2efs_prep_f12.cpp
rfs2efs_prep_f12_shnames.cpp
rfs2efs_prep_f16.cpp
rfs2efs_prep_f16_shnames.cpp
rfs2efs_prep_f32.cpp
rfs2efs_prep_f32_shnames.cpp
runner.c
sblock.cpp
tokenizer.cpp
tokenizer.h
xcompat.cpp

xcompat.h

3.2. Požiadavky na programové prostriedky pri preklade

Na preklad sú potrebné nasledujúce programové prostriedky:

- OS Linux
- utilita *make*
- kompilátor GCC/G++ verzia 3.4.1

3.3. Vlastný preklad

Vykonáme ho rozbalením zdrojových súborov a spustením utility *make*.

4. Náväznosť na iné programové prostriedky

Na prevádzku systému sú potrebné nasledujúce programové prostriedky:

- OS Linux
- WWW server Apache 2.0.50
- PHP 4.3.8
- databázový server MySQL 4.0.20
- kompilátor GCC/G++ verzia 3.4.1

5. Zoznam použitej literatúry

[1] Fox, J.: FAT System Guide. [online] Dostupné na internete: <http://home.freeuk.net/foxy2k/index.htm>.

[2] Microsoft: Microsoft Extensible Firmware Initiative: FAT32 File System Specification.

[3] PHP Documentation. Dostupné na internete: <http://www.php.net/docs.php>.

[4] Cavnar, W.B. – Trenkle, J.M.: N-Gram-Based Text Categorization. [online] Dostupné na internete: <http://www.linguistics.ruhr-uni-bochum.de/~halama/lehre/perl-2004/materialien/cavnar_trenkle_94.pdf>.

[5] [online] Dostupné na internete: <http://people.msoe.edu/~sebern/courses/cs384/papers97/buttron.pdf>.

[6] MySQL Documentation. [online] Dostupné na internete: <http://dev.mysql.com/doc/mysql/en/index.html>.

[7] Answers.com: File Allocation Table. [online] Dostupné na internete: <http://www.answers.com/topic/file-allocation-table>.

[8] OpenLDAP Foundation: Introduction to OpenLDAP Directory Services. [online] Dostupné na internete: <http://www.openldap.org/doc/admin21/intro.html>.

6. Zoznam obrázkov a tabuliek

Zoznam obrázkov

Zoznam tabuliek