Odložišče (Clipboard)

Naum Gjorgjeski

12. januar 2016

1 Splošno o odložišču

Odložišče je programska oprema ki se uporablja za kratkotrajno shranjevanje podatkov in / ali prenos podatkov med dokumenti ali aplikacije s pomočjo kopiraj in prilepi operacij. Običajno je implementiran kot anonimni, začasni medpomnilnik (buffer) podatkov, ponavadi imenovan kot paste buffer. Do tega medpomnilnika lahko dostopa večina aplikacij v okolju preko definiranih programskih vmesnikov. Tipična aplikacija do funkcionalnosti odložišča dostopa tako da uporabnikov vnos preslika na tiste vmesnike (npr. pritisk na tipkovnico, menijsko izbiro itd.).

Standardne operacije na odložišču so (v tem primeru so operacije razložene na delo z besedili, bralec zelo lahko to razširi tudi na druge formate):

Izreži (Cut) Postavi kopijo trenutne izbire na odložišče in pri tem odstrani izbiro od dokumenta. Prejšnja vsebina odložišča je tudi uničena.

Kopiraj (Copy) Postavi kopijo trenutne izbire na odložišče in pri tem dokument ostane nespremenjen. Prejšnja vsebina odložišča je uničena.

Prilepi (Paste) Zamenja trenutno izbiro z vsebino odložišča. Vsebina odložišča ostane nespremenjena.

Večina okoljij podpira enojne transakcije na odložišču. Vsaka operacija Izreži ali Kopiraj prepiše prejšnje vsebine odložišča. Običajno, operacija Prilepi skopira vsebino in pri tem vsebina odložišča ostane na voljo za nadaljnje Prilepi operacije. Glede na to da se odložišče nahaja v RAM-u, ko ugasnemo računalnik se vsebina odložišča zgubi.

2 Upravljavci odložišča (Clipboard managers)

Upravljavci odložišča so aplikacije ki omogočajo uporabniku manipulacijo odložišča. To je program ki dodaja funkcionalnosti osnovnemu odložišču ki ga operacijski sistem ima. Glavna naloga upravljavca odložišča je shranjevanje podatkov ki so skopirani na odložišču na način da omogoča bogatejšo uporabo teh podatkov.

Upravljavci odložišča, poleg osnovnih operacij Izreži, Kopiraj in Prilepi, omogočajo eno ali več funkcionalnosti:

- več medpomnilnikov in možnost združevanja, delitve in spreminjanja njihovih vsebin
- izbira v kateri medpomnilnik naj operaciji Izreži in Kopiraj shranjujejo podatke
- izbira od katerih medpomnilnikih naj operacija Prilepi vzame podatke
- rokovanje formatiranega teksta, tabelarnih podatkov, objektnih podatkov, URL-ji...
- dolgoročno shranjevanje skopiranih podatkov
- iskanje med shranjenimi podatki

3 Odložišče v Windows

Odložišče v Windows je izključno uporabniško usmerjena programska oprema. To pomeni, da okno lahko prenaša podatke v ali iz odložišča le kot odziv na ukaz od uporabnika. Okno ne sme uporabljati odložišče za prenašanje podatkov brez uporabnikovega znanja.

3.1 Formati odložišča (Clipboard formats)

Shranjena informacija na odložišču je lahko v različnih formatih. Tem formatu rečemo format odložišča (clipboard format). Torej okno lahko postavi več kot en objekt na odložišče, vsak objekt pa predstavlja isto informacijo v različnem formatu. Uporabnikom se ni treba zavedati v kakšnem formatu (ali v kakšnih formatih) je neka informacija shranjena na odložišču. Vsak format je določen z nepredznačenim celim številom.

Glede na njihovih lastnostih, Windows ima 5 skupin formatov na odložišču, ki se med seboj ne izključujejo (posamezni format lahko spada v več skupin). To so:

- Standardni formati odložišča (Standard Clipboard Formats)
- Registrirani formati odložišča (Registered Clipboard Formats)
- Privatni formati odložišča (Private Clipboard Formats)
- Večkratni formati odložišča (Multiple Clipboard Formats)
- Sintetizirani formati odložišča (Synthesized Clipboard Formats)

3.1.1 Standardni formati (Standard Clipboard Formats)

Formati odložišča ki so že definirane v sistemu se imenujejo standardni formati odložišča. Med njimi so:

CF_TEXT ANSI tekstovni format. Vsaka vrstica se konča z CR-LF kombinacijo. Znak null signalizira konec teksta.

CF_UNICODETEXT Unicode tekstovni format. Vsaka vrstica se konča z CR-LF kombinacijo. Znak null signalizira konec teksta.

CF_BITMAP Kazalec (ni pravi naslov) na bitmap (za slike).

CF_TIFF Kazalec (ni pravi naslov) na označeno sliko (tagged image).

CF_WAVE Predstavlja zvok v standardnem wave formatu

CF_RIFF Predstavlja kompleksnejši zvok ki ne more biti predstavljen v CF_WAVE standardnem formatu.

CF_HDROP Kazalec (ni pravi naslov) na listo datotek.

CF_PENDATA Predstavlja obliko podatkov za Windows pero (Windows pen data format).

...

3.1.2 Registrirani formati odložišča (Registered Clipboard Formats)

Veliko aplikacij dela s podatki, ki jih ni mogoče prevesti v standardni format odložišča brez izgube informacij. Te aplikacije lahko ustvarijo svoje formate odložišča. Format odložišča ki ga neka aplikacija definira se imenuje registrirani format odložišča. Na primer, če aplikacija za urejanje besedil kopira oblikovano besedilo na odložišče preko standardnega text formata, oblikovanje besedila bi izgubili. Rešitev je da aplikacija registrira nov format odložišča, npr. Rich Text Format (RTF).

Če več kot ena aplikacija registrira format odložišča z enakim imenom, format odložišča je registriran samo enkrat. Na ta način, dve različni aplikaciji si lahko delita podatke preko odložišča z uporabo registriranega formata odložišča.

3.1.3 Privatni formati odložišča (Private Clipboard Formats)

Aplikacija lahko registrira privatni format odložišča z definicijo vrednosti v razponu med CF_PRIVATEFIRST in CF_PRIVATELAST. Aplikacija lahko uporablja privatni format odložišča za takšen format zapisa podatkov ki ga uporablja sama aplikacija in ni potrebno da je ta format registriran v sistemu.

Kazalci na podatkov ki so v privatnih formatih odložišča sistem ne sprosti avtomatično. Če aplikacija uporablja privatne formate odložišča, se mora odzvati na sporočilo WM_DESTROYCLIPBOARD in sprostiti kazalce na podatkov ki so v privatnih formatih.

Ker je glavni cilj odložišča da prenaša podatke med aplikacije, se mogoče ta koncept na prvi pogled zdi nesmiselen. Vendarle odložišče služi tudi za to da prenaša podatke (v formatu ki ga razume ena sama aplikacija) samo znotraj ene aplikacije ali več instanc aplikacije, ki seveda razumejo ta format.

3.1.4 Večkratni formati odložišča (Multiple Clipboard Formats)

Okno lahko postavi več kot en objekt na odložišču, vsak objekt pa predstavlja isto informacijo v različnih formatih odložišča. Ko okno postavi informacijo na odložišču, jo postavi v čim več

formatov. Najprej postavi format ki vsebuje največ informacij, potem pa postavi manj deskriptivne formate. Okno ki potem hoče prilepiti to informacijo običajno pridobi prvi objekt na odložišču ki ga prepozna. Ker so objekti na odložišču v svojem določenem formatu enumerirani v istem vrstnem redu kot so bili postavljeni, prvi objekt na odložišču ki ga aplikacija prepozna je tudi najbolj deskriptivni format na odložišču.

Na primer, recimo da uporabnik kopira formatirano besedilo iz urejevalnika besedil. Urejevalnik besedil najprej postavi podatke v registriranem formatu odložišča, recimo RTF, nato pa postavi podatke na odložišču v manj deskriptivnem formatu, recimo kot navadno besedilo (CF_TEXT). Ko drugo okno prilepi vsebino odložišča, okno pridobi podatke v najbolj deskriptivnem formatu ki ga pozna. Če okno pozna RTF format, dobi podatke kot RTF. Če ne pozna RTF, potem dobi podatke kot navadno besedilo in se formatiranje besedila zgubi.

3.1.5 Sintetizirani formati odložišča (Synthesized Clipboard Formats)

Sistem implicitno pretvarja podatke med določenimi formati odložišča: če okno zahteva podatke v formatu ki ni na odložišču, sistem lahko pretvori nek format ki je na voljo v zahtevanem formatu. Če sistem lahko avtomatsko pretvori določen format v druge formate odložišča, ni potrebe postavljati tudi te formate na odložišče. Na primer, če na odložišču imamo objekt v enem od formatov: CF_TEXT, CF_UNICODETEXT ali CF_OEMTEXT, sistem lahko pretvori objekt v ostalih dveh formatih. Torej dovolj je če bi na odložišče dali le eden od teh treh formatov.

3.2 Lastništvo odložišča (Clipboard ownership)

Od tega odstavka naprej bom včasih govoril o implementacijskih podrobnostih. Pri tem bom uporabljal klice v programskem jeziku C++ na API za odložišče ki ga ima Windows.

Lastnik odložišča je okno povezano z informacijo ki se nahaja na odložišču. Okno postane lastnik odložišča ko postavi podatke na odložišče, oz. natančneje, ko okno pokliče funkcijo EmptyClipboard. Okno ostane lastnik odložišča dokler ni zaprto ali pa drugo okno izprazni odložišča.

Preden se odložišče dejansko izprazni, trenutni lastnik odložišča prejme WM_DESTROYCLIPBOARD sporočilo. En razlog zakaj mora okno sprocesirati to sporočilo je ta da okno ima lahko svoje privatne formate odložišča. Podatke ali pa kazalce na podatkih ki so v privatnem formatu sistem ne sprosti ko se odložišče izprazni. Zato mora trenutni lastnik odložišča le te sprostiti. Drugi razlog leži v odloženem renderiranju, ki ga bom pojasnil v eno od naslednjih poglavjih.

3.3 Operaciji Izreži in Kopiraj

Zdaj pa imamo vse predznanje da razložimo kako dejansko delajo operacije na odložišču. Preden postavi svojo informacijo na odložišču, okno najprej pobriše prejšnjo vsebino odložišča z uporabo funkcije EmptyClipboard. Ta funkcija pošlje WM_DESTROYCLIPBOARD sporočilo prejšnjemu lastniku odložišča, sprosti vire povezane s podatki na odložišču, in dodeli lastništvo odložišča oknu ki je klicalo to funkcijo.

Odložišče je zdaj izpraznjeno, in okno lahko postavi podatke na odložišču v čim več možnih formatov, v vrstnem redu od najbolj deskriptivnem proti najmanj deskriptivnem formatu. Za vsak objekt ki predstavlja določen format, okno kliče funkcijo SetClipboardData, kateri poda format ki ga objekt nosi in kazalec (ni pravi naslov) na objekt. Ta kazalec je lahko NULL, ko gre za odloženo renderiranje.

3.4 Operacija Prilepi

Za pridobitev informacije od odložišča, mora okno najprej ugotoviti kateri format odložišča naj pridobi. Tipično, okno enumerira formate odložišča s klicem na funkcijo EnumClipboardFormats (ta funkcija enumerira formate odložišča v istem vrstnem redu kot so bili postavljeni na odložišče) in uporabi prvi format ki ga prepozna (ta je tudi najbolj deskriptiven). Kot alternativna možnost, okno lahko pridobi informacijo s klicem na funkcijo GetPriorityClipboardFormat. Tej funkciji kot parameter lahko podamo formate odložišča v določenem prioritetnem vrstnem redu in bo funkcija vrnila najbolj deskriptiven format glede na prioritete ki smo jih podali. Če pa okno prepozna le en format odložišča lahko preprosto pokliče funkcijo IsClipboardFormatAvailable, ki pogleda če je ta format na odložišču.

Po določitvi kateri format odložišča naj uporabi, okno kliče funkcijo GetClipboardData. Ta vrne kazalec (ni pravi naslov) na objekt ki vsebuje podatke v zahtevanem formatu.

3.5 Odloženo renderiranje (Delayed rendering)

Ko okno postavi informacije na odložišče, lahko postavi kopije podatkov in da odložišču kazalec na objekt ki vsebuje kopijo. Za zelo velike količine podatkov, ta pristop porabi veliko pomnilnika. Če uporabnik nikoli ne prilepi podatke v drugem programu, bodo podatke še naprej zasedale prostor v pomnilniku dokler jih nekaj drugega ne zamenja.

Temu problemu se izognemo s tehniko imenovano "odloženo renderiranje", kjer okno ne posreduje podatke, dokler jih drugo okno dejansko ne potrebuje. Namesto da bi dali kazalec na podatke, funkciji SetClipboardData lahko kot kazalec na podatke posredujemo NULL.

Ko drugo okno želi pridobiti podatke in kliče funkcijo GetClipboardData, sistem preveri če je kazalec na podatke NULL. Če je, sistem pošlje sporočilo lastniku odložišča da mu posreduje dejanski kazalec na podatke.

Okno ki uporablja odloženo renderiranje mora procesirati 3 sporočila: WM_RENDERFORMAT, WM_RENDERALLFORMATS in WM_DESTROYCLIPBOARD. Sistem pošlje sporočilo WM_RENDERFORMAT ko drugo okno kliče funkcijo GetClipboardData in kot parameter poda zahtevan format. Ko prejme sporočilo WM_RENDERFORMAT, okno mora poklicati funkcijo SetClipboardData in podati kazalec na zahtevan format. Tako drugo okno dobi nazaj kazalec na podatke. Ko drugo okno kliče EmptyClipboard, sistem pošlje WM_DESTROYCLIPBOARD sporočilo in s tem pove da podatke ki so na odložišču ne rabimo več. Prav tukaj se iskaže koristnost te metode. Namreč, zelo verjetno je da velik del teh podatkov ni uporabljen in posledično neuporabljene podatke niso bile niti dostavljene na odložišče.

Zdaj pa se pojavi problem kaj se zgodi če se okno popolnoma zapre. Če se to zgodi, na

odložišču ostanejo kazalci na podatke ki so NULL in če neko okno hoče podatke na odložišču uporabiti, jih dejansko ni! Zato v tem slučaju sistem pošlje sporočilo WM_RENDERALLFORMATS. Tedaj mora okno odstraniti iz odložišča vse objekte ki imajo kazalec na podatke NULL, in za vsak objekt (s pomočjo funkcije SetClipboardData) podati kazalec na podatke. Če potem drugo okno rabi te podatke, jih bo imelo na voljo na odložišču.

4 Odložišče v Ubuntu

Velika razlika od Windowsih je to da Ubuntu nima le en, ampak več "odložišč"ki se imenujejo selekcije. Selekcije v Ubuntu so:

1. Primarna selekcija (Primary selection)
2. Sekundarna selekcija (Secondary selection)
3. Odložiščna selekcija (Clipboard selection)
4. Cut medpomnilniki

Največ se uporabljata primarna in odložiščna selekcija, ki je ekvivalent Windows-ovemu odložišču. Sekundarna selekcija se uporablja bolj redko, v primerih ko ne želimo prepisati vsebine primarne selekcije. Cut medpomnilniki so zastareli in se ne uporabljajo.

Pomembno vlogo za odložišča v Ubuntu ima X okenjski sistem. Ker lahko dva okna upravljajo dve različni aplikaciji, potreben je mehanizem preko katerem komunicirata. Da bi izmenjala podatke preko selekcij lahko uporabljata X strežnik. Protokol za X okenjski sistem vključuje tudi zahtevke in dogodke ki so specifične za izmenjavo vrednosti selekcij, vendar se izmenjava večinoma opravi s pomočjo pošiljanja dogodkov, ki niso specifični za selekcije.

4.1 Pridobivanje lastništvo nad selekcijo (Operaciji Izreži in Kopiraj)

Odjemalec ki hoče pridobiti lastništvo nad določeno selekcijo mora klicati funkcijo SetSelectionOwner, kateri poda za katero selekcijo hoče pridobiti lastništvo, kdo je trenutni lastnik selekcije in čas, ko je to funkcijo poklical (časovni žig). Odjemalci ne morejo neposredno imenovati druge odjemalce, zato trenutni lastnik lahko pridobimo s pomočjo funkcije GetSelectionOwner, kateri podamo za katero selekcijo hočemo pridobiti lastnika. Če klic na funkcijo SetSelectionOwner uspe, strežnik zabeleži odjemalca ki je lastništvo nad določeno selekcijo zahteval kot lastnik te selekcije, za obdobje od časa ki je bil podan (časovni žig) naprej. To je procedura ki se zgodi pri operacijah Izreži in Kopiraj.

4.2 Opuščanje lastništva

Lastnik lahko lastništvo opusti prostovoljno ali kot rezultat akcij nekega drugega odjemalca.

Če lastnik opusti lastništvo prostovoljno, mora narediti zahtevo SetSelectionOwner in kot vrednost za lastnik določiti None, za časovni žig pa isto vrednost kot časovni žig ko je odjemalec prevzel lastništvo. To se najpogosteje zgodi ko se odjemalec zaustavi.

Če nek drug odjemalec uspešno pridobi lastništvo selekcijo, prejšnjemu lastniku se pošlje dogodek SelectionClear.

4.3 Odgovornosti lastnika selekcije (Operacija Prilepi)

Če zahtevnik hoče pridobiti (prilepiti) vrednost selekcije, lastnik selekcije prejme dogodek SelectionRequest. V njem morajo biti specificirani atribute: ciljna selekcija, lastnik te selekcije, cilj (target), lastnina (property), zahtevnik in časovni žig. Selekcija in lastnik ciljne selekcije se morajo sovpadati z vrednosti ki so bile določene v zadnjem SetSelectionOwner zahtevku za to selekcijo. Lastnik selekcije mora preveriti časovni žig zahtevka in časovni žig ko je postal lastnik selekcije. Če je zahtevek poslan s časovnim žigom ki je bolj zgodaj kot časovnega žiga ko je trenutni lastnik postal lastnik selekcije, je SelectionRequest zahtevek zavrnjen in se zahtevniku pošlje dogodek SelectionNotify z lastnino (property) nastavljeno na None. Obstajajo pa tudi bolj napredne lastnike selekcij ki lahko hranijo zgodovino vrednosti selekcije in vračajo vrednost selekcije tudi za obdobja ko dejansko niso bili lastniki selekcije.

Lastnik selekcije glede na cilj (target) se odloči v katerem formatu mora biti selekcija pretvorjena. Nato lastnik selekcije mora postaviti pretvorjene podatke v določeni lastnini (property) na X strežniku. Če to zaradi kateregakoli razloga ne more narediti, zavrne SelectionRequest enako kot smo že omenili v prejšnjem odstavku.

Če so podatki uspešno postavljeni, lastnik to potrdi s pošiljanjem dogodka SelectionNotify X okolju. S tem dogodkom lastnik mora specificirati: zahtevnika, selekcije, cilja, časovnega žiga in lastnine. Vsi ti atributi praviloma imajo enake vrednosti kot vrednosti ki jih je lastnik prejel v SelectionRequest zahtevku (če je lastnina None, se je zgodila napaka).

Če so podatki veliki, lastnik lahko postavi samo del podatkov, oz. pošlje podatke v manjših kosih. Ko so podatki ali del podatkov postavljeni, X strežnik pošlje odjemalcu dogodek PropertyNotify, ki odjemalcu pove da so podatki ali del podatkov na voljo, obenem pa mu pove število bajtov ki jih mora prebrati. Odjemalec nato lahko prebere vrednost selekcije iz lastnine ki je postavljena na X strežniku tako da pošlje GetProperty zahtevo. Ko prebere selekcijo ali del selekcije s pomočjo GetProperty zahteve, poleg podatke ali del podatkov odjemalec v odgovoru dobi tudi podatek o tem koliko bajtov mu je preostalo za prebrati celo selekcijo. Če je ta vrednost večja kot nič, mora odjemalec poslati vsaj še eno GetProperty zahtevo, če pa je vrednost enaka nič, to pomeni da je odjemalec prebral selekcijo. Nato odjemalec pošlje DeleteProperty zahtevo in se lastnina na X strežniku izbriše. Ko odjemalec pobriše lastnino na X strežniku, če odjemalec ni prejel vseh podatkov mora X strežnik lastniku selekcije poslati dogodek PropertyNotify ki mu pove da lahko postavi naslednji del podatkov. Ta proces se izvaja dokler odjemalec ni prejel podatke v celoti.

Omenimo še nekaj podrobnosti. Ko nek drug odjemalec pridobi lastništvo selekcije, prejšnji lastnik prejme dogodek SelectionClear, v katerem so definirane nov lastnik, selekcija in časovni žig (čas ko je nov lastnik prevzel lastništvo nad to selekcijo).

Če se vrednost selekcije spremeni, lastnik pa ostane ist (npr. selektiranje različnih delov nekega besedila), mora lastnik ponovno prevzeti lastništvo selekcije enako kot da sploh ni bil lastnik te selekcije in mora zagotoviti nov časovni žig.