FPCom

Felhasználói kézikönyv

Budapesti Műszaki Egyetem
Híradástechnikai Tanszék

2012.08.30.

Tartalom

1. Listázási alapelvek

A ***FlexPlanet*** munkaterületen található szövegfájlok jóval bonyolultabb összefüggéseket tartalmaznak, mint azt az XPLANET-es elődnél már megszoktuk, ezért feltétlenül szükséges olyan alapvető listák programból történő előállítása, melyek egyébként csak több különböző fájlban való együttes keresgéléssel lennének kinyerhetőek.

A listák formátuma két szempontnak kell megfeleljen:

- Ember által könnyen olvasható és Excelben jól kezelhető legyen.
- Könnyű legyen az esetleges utófeldolgozás (pl. perl, awk).

Kisebb-nagyobb kompromisszumok árán többnyire mindkét feltételnek megfelelő listák állíthatók elő, de természetesen célszerű eleve lehetővé tenni a választást többféle formátum között.

1.1. A listák kimenete

A listázó függvények kimenete egy paraméterként megadható (C++ STL) *ostream* (kimeneti folyam), mely speciálisan tetszőleges fájl is lehet. A lista sorokból áll, a sorok pedig mezőkből. A sorokat mindig sorvég karakter (‘\n’) választja el, a mezők közti elválasztás viszont tetszőleges karakterrel történhet. Alapértelmezésben ez a tabulátor (‘\t’), de pl. a gyakran használt CSV formátumnál az elválasztó karakter vessző (‘,’) vagy pontosvessző (’;’).

A számoknál egyelőre mindig tizedes pontot írunk, de a későbbiekben felmerülhet annak igénye, hogy a magyarban szokásos tizedes vesszős formátum is kiválasztható legyen.

1.2. Oszlopok testre szabása

A listák alapértelmezésben soronként meglehetősen sok információt tartalmaznak, melyek közül a felhasználó a számára érdektelen oszlopok kiírását letilthatja. Ezzel a kimenet kisebb méretű és áttekinthetőbb lesz.

Szintén növeli az áttekinthetőséget, ha az egymás alatti sorokban nem ismételjük meg az előző sorok azonos oszlopának tartalmát, ha az nem változott. Azonban ez az utófeldolgozást nehezítheti, ezért az ismétlés szükség szerint és oszloponként ki- és bekapcsolható.

1.3. Elválasztó- és fejlécsorok

Ha nem a keret vagy a grafikus megjelenítő hívja a listázó függvényeket, akkor célszerű az eredményeket egy vagy több fájlba irányítani. A különböző adatokat tartalmazó táblázatok mindegyikéhez más-más fejlécsort kell nyomtatni, ha pedig egy fájlba több táblázat is kerül, akkor ezeket érdemes legalább egy üres sorral elválasztani egymástól, mert így

áttekinthetőbbek és pl. Excelben is könnyebben kezelhetők. Ezenkívül hasznos lehet, ha tetszőleges megjegyzést is tudunk nyomtatni a táblázatok közé.

1.4. Sorok szűrése

Az áttekinthetőséget nem csak az éppen felesleges oszlopok, hanem a felesleges sorok kiszűrésével is növelni tudjuk. Erre utófeldolgozás jelleggel bármikor lehetőségünk van (pl. Excel AutoFilter), de komolyabb adatmennyiségnél (nagyméretű, összetett hálózatoknál) jobban járunk, ha már a listázás indítása *előtt* kijelöljük az adatok minket érdeklő részhalmazát. Ez az előzetes szűkítés (szűrés) a listázás sebességére is jótékony hatással van, azonban hátránya, hogy ha valami, amit kiszűrtünk, utólag mégis szükségesnek bizonyul, akkor az csak újabb listázással pótolható.

1.5. Halmazok

Bonyolult lekérdezések esetenként csak több lépésben valósíthatóak meg. Ilyenkor valamely lépésben előállt elemek halmazán végzünk további szűréseket, szűkítéseket. Hasznos lehetőség ezért, ha a lekérdezések eredményeit eltárolhatjuk halmazokba, illetve az eltárolt halmazokon különféle szűréseket és halmazműveleteket végezhetünk. A ***FlexPlanet*** ilyen lehetőségek széles skálájával rendelkezik.

A halmazok számának és méretének csak a memória mérete szab határt. A felhasználó szabadon elnevezheti a halmazokat, annyi csak a megkötés, hogy a névnek ’@’ karakterrel kell kezdődnie. A definiált halmazok bármikor egyszerűen lekérdezhetőek illetve törölhetőek is. Az is könnyen kideríthető, hogy egy-egy halmaz éppen mennyi elemet tartalmaz, illetve a benne foglalt elemek különböző formátumokban kilistázhatók.

Két kitüntetett halmaz is van, melyek tartalmát a lekérdezések automatikusan frissítik. Az egyik a „@” nevű, mely mindig az utolsó szűrés eredményét tartalmazza, a másik pedig a „@@” nevű, amely abban különbözik az előzőtől, hogy a node-ok és link-ek hierarchiájának kilistázott elemei is hozzáadódnak. A fentiektől eltekintve ezek a halmazok is ugyanúgy használhatók, mint a többiek, vagyis közvetlenül is módosítható vagy kitörölhető tartalmuk, illetve elemeik listázhatók, szűkíthetők.

1.6. Stringek illeszkedése

A szűrési szabályként gyakran stringekre vonatkozó feltételeket fogalmazunk meg. Ilyen string lehet például az objektumok azonosítója, neve vagy más paramétere. A feltétel pedig kétféle lehet: pontos *egyezés* vagy csak *illeszkedés*. Az utóbbi esetben a feltételben szereplő ’?’ karakter pontosan egy, de bármilyen karakterre illeszkedik, a ’*’ karakter pedig bármilyen és bármennyi karakterre. Továbbá ha pontos egyezés helyett csak illeszkedést használunk, akkor nem teszünk különbséget a kis- és nagybetűk valamint az ékezetes és ékezet nélküli karakterek között.

2. Parancsok

2.1. A kiadható parancsok rövid listája

Az *FPCom* a `help`, illetve bármilyen értelmezhetetlen utasítás hatására kiír egy tömör listát az érvényesen kiadható parancsokról:

```
================================================================================
Msg MSGID {disable | once | enable}
{Read[+] | Write | XPWrite @SET} [PATH] | Clear
echo | {config | ?} | time [ID [0]] | {print | lf} [MSG]
info {add | app | del | mod | upd} ITIP[=INFO] FSET
beult [FSET]
hiemod FILENAME
erase {level... | templ... | FSET}
ospf [/LB[m] /C /A /Z /N /L]* [FSET]
reli [/C /M[N] /V[N] /@SET /m>=M /m[s]<=M /p[s]>=P /pr>P /e /d /s /i]* [FSET]
LIBNAME.dll [FUNCNAME {?[HLP] | [ARG]*}]
FLEXPLAN_COMMAND_FILE | {end | exit | quit | ![OS_COMMAND]}
--------------------------------------------------------------------------------
{List | XLS | SLK | XML | Pos} [FILE [SEP]] | Close
[#] dim
[#] {level | szint} [* | *NEV | nNEV | kKAT | dDIM | D | S | cCSNEV]
[#] {templ | sabl} [* | *ID | nNEV | kKAT | pPOZ | iSNID | sSNNEV | lSLNEV]
[#] *ID
[#] node [* | *ID | [FILT]*]
[#] link [* | *ID | [FILT]*]
[#] site [* | FSET]
info [[?[?] | col] [i(ITIP=INFO)] {net | FSET}]
path {l | g|G | p | r}* LINK_FSET
stat [t|k|l]*[site | p|g|G | i|I|iITIP] [FSET]
sumcap [FSET]
col [templ|level | node|site | xy|clus | tav|{g|e}len | path[g] | iITIP]*
ord [[f]nev|tip|sn|sk|ll|lk|site|sa|sz|SA|SZ|x|y|clus|ca|cz|iITIP]*
mask [[+|-]NN | MASK]* | hm [HMAX] | hc [NN [STR]] | hr [@SET] | hg [@SET]
dos | child
--------------------------------------------------------------------------------
@SET EQ [FSET]
@SET EQ "FILE" [COL]
@SET EQ beult {H | L | [rek] FSET}
@SET EQ {site | device | card | port}
     [i(ID) n(NEV) s(SID) m(@PARENTS) g(@CHILDREN) x(X) y(Y) c(CLUS)]*
@SET EQ {graf | mpx | log | dem | res}
     [i(ID) n(NEV) l(SZINT) {a|z}(NID) g(LEN) r($GRAF)]*
@SET :[:] [n(NEV) s(SID) x(X) y(Y) c(CLUS) l(SZINT) g(LEN) h(ID) p(POZ)]*
[#] @SET [FILT]*
[#] set [clear]
EQ: = += -= *=
FSET: *ID | {node | link | all | @SET} [FILT]*
FILT: [!] t(TIP) i(ITIP=INFO) {sa|SA | sz|SZ | s|S}([NNEV | @SET]) NDX
      {nn|NN}(NNEV) sn(SNEV) sk(SKAT) xy([R]) c(CLUS)
      {ln|LN}(LNEV) ll(SZINT) lk(SZKAT) {p|g|G}(PATH)
--------------------------------------------------------------------------------
$GRAF [GSW]* [[+]= [^]* [FSET]]
GSW: /f /a /z /g /t /Hop /Len /Cap /A[A] /Z[Z] /RC /R1[n] /R2[n] /p
graf [clear]
================================================================================
```

A lista láthatóan csak egy szintaktikai emlékeztető, az egyes parancsok részletes működését a következő fejezetekben tárgyaljuk. A kulcsszavak (melyeket a megadott formában kell begépelni) néhány kivételtől eltekintve kisbetűsek. A csupa nagybetűvel írt rövidítések helyett

azok jelentésének megfelelő tartalmat kell írni. Pl. a `@SET` helyett egy konkrét halmaznevet kell megadni, vagy az `x(X)` koordináta megadás egy tényleges alakja: `x(540.12)`.

A `//` kezdetű sorokat valamint a `/*` és a `*/` kezdetű és a közéjük zárt sorokat nem értelmezi a program, azokat megjegyzésnek tekinti.

2.2. Hibaüzenetek kezelése

```
Msg MSGID {disable | once | enable}
```

Hibaüzenet letiltása (`disable`) illetve engedélyezése (`enable`). A `once` kapcsoló azt jelenti, hogy az adott hiba többszörös előfordulása esetén csak az első alkalommal jelenik meg hibaüzenet.

Példa:

```
Msg -OPENR disable
```

A hibaüzenetek a **`network.ini`** fájlban is konfigurálhatók az `[MSG]` szekcióban:

```
[MSG]
+TIMETXT= disable
-OPENR= disable
```

2.3. Munkaterület beolvasása és kiírása

```
{Read[+] | Write | XPWrite @SET} [PATH] | Clear
```

A **`DEMO`** könyvtárban található munkaterület beolvasása:

```
Read DEMO
```

A parancs először törli a már beolvasott munkaterületet a memóriából, ha van ilyen. Ha beolvasás nélkül akarjuk törölni a memóriát, akkor használjuk a `Clear` parancsot; ha pedig a memória törlése nélkül akarunk új adatokat beolvasni (hozzáfűzni a memóriában lévőkhöz), akkor a `Read+` parancsot. Pl. új adatok hozzáfűzése a már beolvasottakhoz a ..**`\EXT\DEMO`** könyvtárból:

```
Read+ ..\EXT\DEMO
```

A memóriában lévő adatok kiírása a **`C:\FlexPlan\NEW\DEMO2`** könyvtárba:

```
Write C:\FlexPlan\NEW\DEMO2
```

A `@XP` halmaz elemeinek és azok hierarchiájának kiírása a **`DEMO\XPLANET`** könyvtárba XPLANET-es formátumban:

```
XPWrite @XP DEMO\XPLANET
```

A parancs kiadása előtt természetesen a `@XP` halmazt megfelelően fel kell tölteni, pl. a későbbiekben bemutatásra kerülő szűrő-parancsokkal. A kiírás a kiírt hálózatelemekhez hozzáad egy `XPW=OK` info-t, illetve azokat az elemeket nem írja ki, melyeknek már van `XPW` típusú info-ja. Emiatt kiírás előtt általában célszerű törölni az `XPW` info-kat.

Ha a beolvasó vagy kiíró parancsokhoz nem adunk meg könyvtárnevet, akkor a parancs az aktuális könyvtárra vonatkozik.

2.4. Kimeneti listák megnyitása és lezárása

```
{List | XLS | SLK | XML | Pos} [FILE [SEP]] | Close
```

A listázó parancsokhoz formátumot és kimeneti fájlt rendelhetünk Egyszerre legfeljebb egy kimenet lehet megnyitva, ami lezáródik a következő kimenet megnyitása előtt illetve a `Close` parancs kiadásakor. Ha nincs megnyitott kimenet, akkor a legtöbb listázó függvény a standard outputra ír valamilyen egyszerű formátumban, pl. csak a hálózatelemek azonosítóját felsorolva.

`List`: Egyszerű, nem hierarchikus lista formátum.

`XLS`: Részletesebb, hierarchikus lista formátum.

`SLK`: Az `XLS` formátum SILK változata, melyet egyszerű szövegszerkesztőkkel nehezebb olvasni, azonban Excel-ben kevesebb probléma adódik a mezőelválasztó karakterekkel és a tizedesponttal / -vesszővel.

`XML`: Linkek XML formátumú, hierarchikus listázása. Egyéb listák esetén megegyezik az egyszerű `List` formátummal.

`Pos`: Linkek foglalt és szabad pozícióinak listázása. (A foglalt pozíciókat az utolsó `beult` parancs eredményéből keresi ki a program, lásd 2.10. Beültetés, 16. oldal.) Egyéb listák esetén megegyezik az egyszerű `List` formátummal.

`FILE`: A kimeneti lista-fájl neve. Ha nem adjuk meg, vagy ha csak egy ’-’ karakter, akkor a standard output lesz a kimenet, ahova azonban a kiválasztott formátumban íródnak a listák.

`SEP`: Mezőelválasztó karakter. Ha nem adjuk meg, akkor a tabulátor. XML formátum esetén nincs jelentősége.

2.5. Halmazok és szűrők

A halmazoknak értéket adni az alábbi formátumú parancsokkal lehet:

```
@SET EQ [FSET]
```

`EQ` a következő értékadó operátorok egyike: egyszerű értékadás (`=`), unió (`+=`), különbség (`-=`), metszet (`*=`). Pl. ha a `@A` halmazhoz hozzá akarjuk adni a `@B` halmaz elemeit (természetesen unió értelemben), akkor a `@A+=@B` parancsot kell kiadni. Az értékadás jobb oldalán nem csak egy halmaznév állhat, hanem egy `FSET` kifejezés, amely szintaktikája:

```
*ID | {node | link | all | @SET} [FILT]*
```

`*ID`: Node vagy link ID, mely előtt a ’*’ karaktert ki kell írni, pl.: `*M12345`.

`node`: Az összes node.

`link`: Az összes link.

`all`: Az összes node és az összes link.

`@SET`: Halmaznév.

`FILT`: A `node` és a `link` kulcsszavak illetve egy halmaznév után opcionálisan egy vagy több szűrőfeltételt is megadhatunk, melynek jelentése: „Azok a node-ok / link-ek / halmazelemek, melyekre a megadott feltételek **mindegyike** teljesül.” Az egyes feltételeket szóközökkel kell egymástól elválasztani. Bármelyik feltétel kódja előtt használhatjuk a ’!’ karaktert, ha azokat az elemeket keressük, melyekre az adott feltétel **nem** teljesül. A felkiáltójelet és a kódot egybe kell írni, köztük szóköz sem lehet.

Lehetséges szűrőfeltételek:

`t(TIP)`: Típus (`site, fsite, device, card, port, log, dem, mpx, res, graf, …`).

`i(ITIP=INFO)`: Info (típus=érték). A típust és az értéket egy ’=’ karakterrel kell elválasztani, és mindkét kifejezésben szerepelhetnek ’?’ és ’*’ karakterek.

`nn(NNEV)`: Node neve. Ha az `nn` helyett a nagybetűs `NN` alakot használjuk, akkor a megadott névnek pontosan egyeznie kell; tehát lényeges a kis- és nagybetűk valamint az ékezetes karakterek közti különbség, és nincs speciális jelentése a ’?’ illetve ’*’ karaktereknek.

`sn(SNEV)`: Sablon (node-template) neve.

`sk(SKAT)`: Sablon kategóriája.

`xy()`: Ha ebben a formában, érték nélkül használjuk, akkor a feltétel azokra a telephelyekre teljesül, melyeknek legalább egyik koordinátája nullától különböző.

`xy(R)`: A kifejezés azt jelenti, hogy azok helyett a node-ok helyett, melyekre a többi feltétel teljesül, a node telephelyének `R` sugarú körzetében található telephelyek kerülnek be a halmazba. Az invertált forma (`!xy(R)`) használatakor az `R` sugarú körön kívül eső telephelyek kerülnek be a halmazba.

`c(CLUS)`: Cluster (körzet). Csak telephelyekre (site) értelmezett.

`ln(LNEV)`: Link neve. Ha az `ln` helyett a nagybetűs `LN` alakot használjuk, akkor a megadott névnek pontosan kell egyeznie.

`ll(SZINT)`: Szintnév.

`lk(SZKAT)`: Szint kategóriája.

`p(SZKAT)`: Nyomvonal ellenőrzés eredménye (`OK, EXT, REKH, ERR, NOHIE, …`; lásd `path p` parancs.)

`g(SZKAT)`: Nyomvonal és hossz ellenőrzés eredménye. (Lásd `path g` parancs.)

`G(SZKAT)`: Hossz ellenőrzés eredménye. (Lásd `path G` parancs.)

`sa(NNEV)`: Link kezdőpontja (’A’ oldali telephely). Ha az `sa` helyett a nagybetűs `SA` alakot használjuk, akkor a megadott kifejezésnek a kezdőpont teljes nevére (`SITE::DEVICE::CARD::PORT`) kell illeszkednie.

`sz(NNEV)`: Link végpontja (’Z’ oldali telephely). Ha az `sz` helyett a nagybetűs `SZ` alakot használjuk, akkor a megadott kifejezésnek a végpont teljes nevére kell illeszkednie.

`s(NNEV)`: Node telephelye vagy link valamelyik végpontja. Ha az `s` helyett a nagybetűs `S` alakot használjuk, akkor a megadott kifejezésnek a pont teljes nevére kell illeszkednie.

`sa(@SET)`: A feltétel azokra a link-ekere teljesül, melyek ’A’ oldali telephelye benne van a megadott halmazban. Ha az `sa` helyett a nagybetűs `SA` alakot használjuk, akkor a kezdőpont felmenő node-hierarchiájának valamelyik tagja kell, hogy a halmaz eleme legyen. Ez a feltétel csak egyszer szerepelhet a feltétel listában.

`sz(@SET)`: Hasonló, mint az `sa/SA` feltétel, de a linkek ’Z’ oldalára vonatkozik.

`s(@SET)`: Hasonló, mint az `sa/SA` feltétel, de elég, ha a linkek *bármelyik* végpontja a halmazban van, illetve node-okra is értelemszerűen működik.

`sa()`: Ha ebben a formában, érték nélkül használjuk, akkor a kifejezés azt jelenti, hogy azok helyett a link-ek helyett, melyekre a többi feltétel teljesül, a kezdőponti telephely kerül be a halmazba. A nagybetűs `SA` alak esetén a kezdőpont teljes felmenő hierarchiája kerül be a halmazba. Ennél a feltételnél az invertálásnak nincs hatása, tehát az `sa()` és `!sa()` ugyanazt jelenti.

`sz()`: Hasonló, mint az `sa()/SA()` feltétel, de a linkek ’Z’ oldalára vonatkozik.

`s()`: Hasonló, mint az `sa()/SA()` feltétel, de linkek helyett node-okra vonatkozik.

`NDX`: Ha a szűrőfeltétel egynél több elemre is teljesül, de mi közülük csak egyet akarunk példaként kiválasztani, akkor a feltételek között soroljunk fel egy pozitív egész indexet. A feltételt teljesítő elemek indexelése 1-től egyesével növekszik, de pontos sorrendjük a belső ábrázolástól függ, a felhasználó számára véletlenszerűnek tekintendő.

Ha a feltételek között felsorolunk olyat is, amelyiket a program nem tudja értelmezni, akkor ez nem jelent szintaktikai hibát, azonban a feltétel biztosan nem teljesül egyetlen elemre sem.

Az egyes feltételek azonosítója után kell megadni a hozzá tartozó értéket, pl. zárójelek között. Nem csak a () zárójel-pár használható, hanem a [], {} és <> is, viszont a feltétel azonosítója és a nyitó zárójel között nem lehet szóköz vagy egyéb karakter. Ha a feltétel azonosítóját nem nyitó zárójel követi, akkor a program azt a karakter tekinti határoló karakternek. Sok esetben a legkényelmesebb, ha erre a célra a szóközt használjuk, de mindenképpen olyan karaktert kell választani, amely nem szerepel a kifejezésben.

Egy halmazt kitörölhetünk a memóriából a `@SET=` alakú paranccsal, tehát ha az egyszerű értékadás jobb oldalát üresen hagyjuk.

Egy halmaz elemei, vagy azok egy részhalmaza listázható az aktuális kimenetre az alábbi paranccsal:

```
@SET [FILT]*
```

Ha csak a (rész)halmaz elemszámára vagyunk kíváncsiak, akkor a parancsot kezdjük a ’#’ karakterrel:

```
# @SET [FILT]*
```

Az összes halmaz nevének és méretének listázása:

```
set
```

A halmazok darabszámának lekérdezése:

```
# set
```

Az összes halmaz törlése a memóriából (körültekintően használjuk!):

```
set clear
```

2.5.1. Példák a halmazok és szűrők használatára

Gyűjtsük ki az `@ext` halmazba azokat a berendezéseket, melyeknek van „ext” info-ja:

```
@ext= node t device i ext=*
```

Listázzuk azokat a telephelyeket, melyek neve az „ismeretlen” karaktersorozattal kezdődik:

```
node t site nn ismeretlen*
```

Rakjuk bele a `@noxy` halmazba azokat a gráf-végpontokat, melyeknek nincs koordinátája:

```
@noxy= link t graf sa() sz()
@noxy= @noxy !xy()
```

Listázzunk ki a fenti halmazból két találomra kiválasztott budapesti pontot:

```
@noxy nn budapest* 1
@noxy nn budapest* 2
```

Írassuk ki, hogy mennyi olyan telephely van, aminek a nevében szerepel szóköz karakter:

```
# node t site nn(* *)
```

2.5.2. Halmazok beolvasása fájlból

```
@SET EQ "FILE" [COL]
```

A parancs beolvassa az idézőjelek közé írt nevű fájl megadott oszlopában szereplő kulcsokat, és az így azonosított elemekkel végzi el az `EQ` műveletet. Az oszlopokat tabulátor karakterek határolják, számozásuk 1-től indul, de ha nem adjuk meg, akkor az első oszlopot olvassa be. A ’#’ karakterrel kezdődő sorok megjegyzések, ezeket a beolvasás átugorja. Ha valamely sor megadott oszlopában nem érvényes node vagy link azonosító található (és a mező nem is üres), akkor hibaüzenetet kapunk.

2.6. Új hálózatelemek felvétele

Új hálózatelemet úgy tudunk létrehozni, ha egyúttal halmazba is rakjuk:

```
@SET EQ {site | device | card | port | graf | mpx | log | dem | res}
```

Ez azért szükséges, hogy később „megtaláljuk”, hiszen a fenti formában automatikusan generálódik azonosító és név az új elemhez, tehát nem tudhatjuk, hogy hogyan hivatkozhatunk rá.

Ha mi akarjuk megadni az azonosítót vagy a nevet, akkor a parancsot ki kell egészíteni az `i(ID)` vagy `n(NEV)` paraméterekkel is. A szűrőfeltételekhez hasonlóan itt is használhatunk a () zárójelektől eltérő határoló karaktereket.

Nézzünk néhány példát arra, hogy hogyan tudunk új telephelyeket és gráféleket felvenni, melyek közül nem mindegyiknek adjuk meg az azonosítóját és a nevét. Az új elemeket a `@new` halmazban tároljuk el:

```
@new= site n Ujtelep
@new+= site i UJ12345 n Ujtelep/2
@new+= graf i G1000
```

Ha véletlenül olyan azonosítót választottunk, amelyik már foglalt, akkor a program helyette automatikusan egyedi azonosítót generál. A névnek azonban nem kell egyedinek lennie.

Node-ok esetén megadható további paraméterek:

`s(SID)`: Sablon azonosító. (Ha nincs ilyen sablon, a program hibát üzen.)

`m(@PARENTS)`: Szülő node-ok halmaza. Egy node-nak általában csak egy szülője van, pl. egy port-nak egy kártya, egy kártyának egy berendezés, egy berendezésnek egy telephely.

`g(@CHILDREN)`: Gyerek node-ok halmaza. Egy node-nak több gyereke is lehet, pl. egy berendezésnek a kártyái, egy kártyának a port-jai.

`x(X)`: Telephely X koordinátája.

`y(Y)`: Telephely Y koordinátája.

`c(CLUS)`: Telephely cluster-e.

Link-ek esetén megadható további paraméterek:

`l(SZINT)`: Link szintje.

`a(NID)`: 'A' oldali node azonosító.

`z(NID)`: 'Z' oldali node azonosító.

`g(LEN)`: Gráf él hossza.

`r($GRAF)`: Link nyomvonala. (Részletesen lásd 2.13. Gráfok, 20. oldal.)

2.7. Hálózatelemek törlése

```
erase level...
```

A parancs végérvényesen és visszavonhatatlanul eltávolítja a hálózatból a kijelölt szintek közül azokat, melyek egyik linkhez sem tartoznak. A kijelöléssel részletesen a 2.14.2. Szintek fejezet foglalkozik, lásd a 23. oldalon. Ha a `level` kulcsszó után nem írunk semmit, akkor a parancs minden használaton kívüli szintet töröl.

```
erase templ...
```

A parancs végérvényesen és visszavonhatatlanul eltávolítja a hálózatból a kijelölt sablonok közül azokat, melyek egyik node-hoz sem tartoznak. A kijelöléssel részletesen a

2.14.3. Sablonok (node template-ek) fejezet foglalkozik, lásd a 23. oldalon. Ha a `templ` kulcsszó után nem írunk semmit, akkor a parancs minden használaton kívüli sablont töröl.

```
erase FSET
```

A parancs végérvényesen és visszavonhatatlanul eltávolítja a hálózatból az `FSET`-ben szereplő elemeket.

A (megmaradó) link-ek hierarchiájából is kiveszi a törölt elemekre vonatkozó hivatkozásokat, és egy `Clus=n` info-val jelzi, hogy hány hivatkozást távolított el a hierarchiából (`n` pozitív egész szám).

Az `FSET`-ben lévő link-ek közül csak azokat törli ki, amelyeknek a fenti lépés után teljesen üres lesz a kifejtése, a többit viszont megjelöli egy `Clus=Res` info-val. Ezzel biztosított, hogy foglaltságokkal kapcsolatban nem veszítünk információt.

Ha egy link-et mindenképpen törölni akarunk, akkor az `erase` parancs kiadása előtt töröljük a hierarchiáját. (Lásd 2.8. Hálózatelemek módosítása, 13. oldal.) Pl. ha a `@DEL` halmazban vannak a törlendő link-ek, melyeket foglaltságként sem akarunk megőrizni, akkor ezt a következő parancsokkal tehetjük meg:

```
@DEL :: h()
erase @DEL
```

2.8. Hálózatelemek módosítása

Ha módosítani akarjuk egy hálózatelem paramétereit, akkor előtte egy halmazba kell rakni, majd az alábbi formátumú paranccsal lehet elvégezni a módosítás(oka)t:

```
@SET : [n(NEV) s(SID) x(X) y(Y) c(CLUS) l(SZINT) g(LEN) h(ID) p(POZ)]*
```

Ha a halmaz egynél több elemet tartalmaz, akkor hibaüzenetet kapunk. Előfordulhat azonban, hogy szándékosan akarunk egyszerre több elemet azonosan módosítani. Ilyenkor használjuk a halmaznév után a `:` operátor helyett a `::` operátort.

A módosítható paraméterek értékét a szűrőfeltételekhez hasonlóan itt is határolhatjuk a () zárójelektől eltérő karakterekkel.

`n(NEV)`: Node vagy link neve. A `NEV`-ben szerepelhetnek speciális kifejezések, melyeket a program az alábbiak szerint lecserél:

- `{nn}` Node eredeti neve
- `{ln}` Link eredeti neve
- `{sa}` Link ’A’ oldali telephelyének neve.
- `{sz}` Link ’Z’ oldali telephelyének neve.
- `{ll}` Link szintjének neve.
- `{++}` Növekvő sorszámozás (minden parancssorban újraindul 1-től).

`s(SID)`: Sablon (node template) azonosító.

`x(X)`: Telephely X koordinátája.

`y(Y)`: Telephely Y koordinátája.

`c(CLUS)`: Telephely cluster-e.

`l(SZINT)`: Link szintje.

`h(ID)`: Link közvetlen szakaszpontjának (node) vagy szakaszának (link) azonosítója. (Itt az `FSET` szintaktikától eltérően nem kell '*' karaktert írni az azonosító elé.) Több `h(ID)` elem is megadható, ezek fogják a link új hierarchiáját alkotni, egyszerű soros blokként, a megadás sorrendjében. A link eredeti, módosítás előtti hierarchiája törlődik.

`p(POZ)`: Az előző `h(ID)`–vel adott linkre vonatkozó beültetési pozíció (1-től kezdődő) indexe. Ha nem link `ID`-jét követi, akkor hibaüzenetet kapunk.

Példaként tegyük fel, hogy a `@M` halmaz egyetlen eleme egy link, melynek nevét, szintjét és hierarchiáját szeretnénk módosítani. A kezdőpont azonosítója legyen `S1`, a végponté `S2`, és a kettő között egyetlen szakasz, az `M15` link, melynek a 2. pozíciójába akarjuk beültetni a módosított linket:

```
@M : n(Új név) l KÁBEL h S1 h M15 p 2 h S2
```

2.8.1. Linkek hierarchiájának (kifejtésének) módosítása

Az előzőekben láttunk egy módszert linkek hierarchiájának módosítására, azonban ha több, meglévő hierarchiájú linken kell kisebb módosításokat végrehajtani és esetleg új nyomvonalat is tervezni, akkor kényelmesebb a következő parancs használata:

```
hiemod FILENAME
```

Ez a parancs beolvassa a `FILENAME` nevű fájlt, mely sorai tabulátorokkal elválasztott mezőkből állnak, és végrehajtja a kijelölt módosításokat. Nézzük meg, hogy hogyan lehet az előző példában látotthoz hasonló módosítást ebben a formában leírni.

Tegyük fel, hogy a módosítani kívánt link azonosítója `M999`. (Fent ez volt a `@M` halmazban, akkor nem kellett ismerni az azonosítót.) Az első sor 1. oszlopában '0' karakter jelzi, hogy a 2. oszlopban van a módosítandó link azonosítója. A további oszlopokban lévő adatok lényegtelenek, azokat a program megjegyzésnek tekinti.

A következő sorok '1' karakterrel kezdődnek, ami azt jelenti, hogy ezek alkotják a hierarchia elemeit. A 2. oszlopban van az elemek azonosítója, a 3. oszlop megjegyzés, a 4. oszlop pedig a beültetési pozíció. Ez utóbbi a pozíció indexe, ha '#' karakterrel kezdődik, egyébként viszont a pozíció neve.

```
0         M999                BERELT LINK
1         S1                  BUDAPEST
1         M15                 BUDAPEST-MISKOLC          #2
$KABEL    node nn MISKOLC     node nn SZERENCS
1         S3                  SZERENCS
```

A negyedik sor olyan utasítást tartalmaz, melyet az előző fejezetben bemutatott egyszerűbb paranccsal nem tudunk helyettesíteni. Ez az utasítás utat keres a `$KABEL` nevű gráfon `MISKOLC` és `SZERENCS` között, és a megtalált nyomvonal szakaszai kerülnek bele az `M999`-es link hierarchiájába. A `$KABEL` gráf csomópontjait és éleit, valamint az élek súlyait még a `hiemod` parancs kiadása előtt meg kell adni a megfelelő gráf parancsokkal (lásd 2.13. Gráfok, 20. oldal). Egy link hierarchiáját több ponton is módosíthatjuk ehhez hasonlóan, sőt, az egyes útkeresések akár különböző gráfokon is történhetnek.

Egy fájlban több link hierarchiáját is módosíthatjuk, ha ezeket egyenként egy-egy ’0’ kezdetű vezérsorban megadjuk. Az ’1’ kezdetű sorok mindig az őket megelőző utolsó vezérsor linkjére vonatkoznak. A könnyebb olvashatóság érdekében tagolhatjuk a leírást üres sorokkal, illetve megjegyzéseket írhatunk a sor elején alló ’#’ karakter után.

Az ismertetett fájlformátum azért hasznos, mert a későbbiekben látni fogjuk, hogy a linkek meglévő hierarchiája könnyen listázható ebben a formában, majd egyszerűen módosítható szövegszerkesztővel vagy Excel-ben.

2.9. Hálózatelemek kiegészítő paraméterei

2.9.1. Hálózatelemek info-inak lekérdezése

```
info [[?[?] | col] [i(ITIP=INFO)] {net | FSET}]
```

`?`: Csak az info típusok listázása.

`??`: Info típus és érték párok listázása.

`col`: Az info típusokat hozzáadja az aktuális `col` lista (lásd 2.14.11. Konfigurálható oszlopok, 26. oldal) végére.

`i(ITIP=INFO)`: Csak a szűrő-feltételnek megfelelő info-k listázása.

`net`: A hálózat info-inak listázása.

`FSET`: A kijelölt elemek info-inak listázása.

Ha hiányzik a `?`, `??` vagy `col` kapcsoló, akkor nem csak összesítő listát kapunk, hanem a megadott halmaz minden elemére egyenként listázza a program, hogy milyen (a szűrő-feltételnek megfelelő) info-i vannak.

2.9.2. Hálózatelemek info-inak módosítása

```
info {add | app | del | mod | upd} ITIP[=INFO] FSET
```

A parancs az `FSET`-ben definiált elemekre hajtja végre a kijelölt módosítást.

`ITIP`: Info típus.

`INFO`: Info érték. Ha nem adjuk meg, akkor üres string lesz. Néhány speciális kifejezést a program az alábbiak szerint helyettesít:

- `ftav`: A végpontok közti földrajzi (légvonalban mért, euklideszi) távolság.
- `glen`: A link (gráf éleken mért) hossza.
- `randN`: A [0, N] intervallumba eső egész szám. Ha N hiányzik vagy 0, akkor a [0, 1] intervallumba eső valós szám.
- `iNEV`: A `NEV` nevű info értéke. Ha nincs ilyen info, akkor üres string.

Kétoperandusú műveleteket is megadhatunk a következő műveleti jelekkel: `< > + - * / ^ &`

`tav^d=(tav*d)/(1+(tav-1)*d)`

Pl. ha az egységnyi hosszúságú kábel DTR-je `3.5e-5`, akkor egy `glen` hosszúságú kábel DTR-jét a `glen^3.5e-5` kifejezéssel írhatjuk le.

`iA&iB` Az `A` és a `B` info összefűzése (concatenate).

A műveleti jelek közül csak az első viselkedik operátorként. Ezt kell kihasználnunk, ha olyan stringet akarunk megadni, mely tartalmaz műveleti jelet, pld. `info upd DTR=&1e-6 @`.

`add`: Az elemek info-listájának elejére illeszt egy `ITIP=INFO` info-t.

`app`: Az elemek info-listájának végére illeszt egy `ITIP=INFO` info-t.

`del`: Az elemek info-listájából kitörli azokat az info-kat, melyekre illeszkedik az `ITIP=INFO` minta.

`mod`: Ha egy elem info-listájában van `ITIP` típusú info, azok közül az első értékét `INFO`-ra módosítja.

`upd`: Ha az elemnek van már `ITIP` típusú info-ja, akkor `mod`, egyébként `app`.

2.10. Beültetés

A belső ábrázolás a link-ek hierarchiáját felülről lefelé tárolja, vagyis csak azt írja le, hogy egy link-nek milyen szakaszai vannak, azaz milyen link-ekbe és node-okba van beültetve. A fordított irányú lekérdezés, tehát hogy egy adott link-be vagy node-ba mely linkek nyalábolódnak, ez alapján is megoldható, azonban rendkívül lassú lenne. A `beult` paranccsal felépíthetünk egy olyan belső adatszerkezetet, mely segítségével a beültetések is gyorsan kikereshetőkké válnak:

```
beult [FSET]
```

A parancs lefutása után az `FSET`–ben szereplő link-ek hierarchiájában tudunk „felfelé”, tehát a beültetés irányában is keresni. Ha elhagyjuk az `FSET` kifejezést, akkor minden link hierarchiáját feltérképezi a program. Egyszerre csak egy beültetést gyorsító adatszerkezet lehet a memóriában, ami azt jelenti, hogy mindig csak az utolsó `beult` parancs eredményét tudjuk használni. Általában célszerű a beolvasás után rögtön kiadni a paraméter nélküli változatot, ami ugyan nagy hálózatoknál lassú, azonban ezután mindegyik link hierarchiájában tudunk keresni.

Lekérdezési lehetőségek a `beult [FSET1]` parancs futtatását követően:

`@SET EQ beult H`: A beültetés „felső” elemei (`FSET1` linkjei).

`@SET EQ beult L`: A beültetés „alsó” elemei (`FSET1` linkjeinek kifejtésében szereplő link-ek és node-ok).

`@SET EQ beult FSET2`: Az `FSET2` elemekbe beültetett linkek. A `beult` kulcsszó és `FSET2` közé beírhatjuk a `rek` kulcsszót is, ha nem csak a közvetlenül beültetett link-ekre vagyunk kíváncsiak, hanem („rekurzívan”) a magasabb szintűekre is.

2.11. Megbízhatósági analízis

```
reli [/C /M[N] /V[N] /@SET /m>=M /m[s]<=M /p[s]>=P /pr>P /e /d /s /i]* [FSET]
```

A megbízhatósági analízis során a hálózatelemek két fontos csoportját kell meghatároznunk: ez *esemény*eket és az *igény*eket. Minden elem tartozhat bármelyik csoportba, akár mindkettőbe is vagy egyikbe sem.

2.11.1. Elemi események

A hálózatelemek időnként meghibásodnak, és ilyenkor kijavításukig nem tudják ellátni feladatukat. Ha egy elem hibátlanul működik, akkor azt mondjuk, hogy „up-state”, ha teljesen használhatatlan, akkor pedig „down-state”. A működőképességet szokás jelölni egy *0* és *1* közé eső valós számmal is, ahol a *0* a *down*, az *1* az *up*, és pl. a *0.8* azt jelenti, hogy az elem 80%-ban használható.

Az elemi események egymástól függetlenül következhetnek be, és a ***FlexPlanet***-ben használt modell szerint minden ilyen esemény pontosan egy hálózatelemhez rendelhető. Az esemény fontos jellemzője a valószínűsége, vagyis a hozzá rendelt elem DTR-je (Down Time Ratio), ami annak a valószínűsége, hogy egy tetszőleges pillanatban az elem *down* állapotban van. Egy hálózatelemhez legfeljebb egy elemi eseményt kapcsolhatunk, s ha ezt megtesszük, akkor ezzel a hálózatelemet az események csoportjába soroljuk.

Egy adott hálózati állapot egyértelműen definiálható, ha felsoroljuk a bekövetkezett eseményeket. Az ezekhez tartozó elemek működőképessége *0*. Ha *N* elemi eseményünk van, akkor ezek egy 2^N méretű állapotteret feszítenek ki, vagyis a hálózat ennyiféle (nem feltétlenül különböző) állapotban lehet.

Az analízis eseményeit a

```
reli FSET
```

paranccsal tudjuk megadni. Előtte azonban ne felejtsük el beállítani az FSET-ben szereplő elemek DTR info-ját! Ha FSET üres halmaz, akkor nincsenek események, az állapottér egyetlen (2^0) állapotot tartalmaz, a hibamenteset.

2.11.2. Események összevonása

Az *egy adott igényhalmazra nézve sorosan viselkedő* elemi eseményeket összevonhatjuk egyetlen eseménnyé az alábbi paranccsal:

```
reli /C FSET
```

Az igényhalmazt FSET definiálja. Az E1 és E2 elemi esemény akkor viselkedik sorosan FSET igényeire nézve, ha külön-külön és együttesen is az igények ugyanazon részhalmazán okoznak *1*-nél kisebb működőképességet:

Hiba{E1} ≡ Hiba{E2} ≡ Hiba{E1+E2} ⇒ E1 és E2 összevonható

Az összevonás az igényhalmazban szereplő igényeket kiértékeli minden egyszeres hibaállapotban, de csak azokban; és minden olyan eseményt összevon, melyek önmagukban bekövetkezve ugyanazt a hibahatást eredményezik, de azt már nem ellenőrzi, hogy ez a

többszörös hibák esetén is teljesül-e. **Ennek eldöntése a felhasználó felelőssége!** Általános szabályként javasolható, hogy az összevonásnál használt igényhalmaz legyen a legmagasabb szint a hálózatban, ahol még nincs védelem. Itt ugyanis biztosak lehetünk benne, hogy ha `E1` és `E2` kiesése egyenként ugyanazt a részhalmazt ejti ki, akkor a két esemény együttes hatása is ezzel azonos lesz. Védelem esetén ez már nem feltétlenül igaz, hiszen pl. 1+1-es elvezetésnél az egyszeres kábelhibák nem okoznak kiesést, a kétszeres hibák hatására azonban már előfordulhat teljesítménycsökkenés:

Hiba{E1} ≡ Hiba{E2} < Hiba{E1+E2}

Egy összevont esemény neve a benne összevont események közül az elsőnek a neve lesz, a valószínűsége pedig az összevont események DTR-jének soros eredője. Ez azonban a hálózatba nem íródik vissza (egyik hálózatelem neve vagy info paramétere sem fog megváltozni), kizárólag a következő `reli` parancsokra lesz hatással, azokra is csak addig, amíg nem definiálunk új eseményhalmazt. Az összevonások nyomon követhetőek, ha a `/C` kapcsoló után megadjuk a `/e /d /s` kapcsolókat is. Az összevont halmaz lekérdezhető pl. a `reli /M0 /e` paranccsal.

2.11.3. Állapotok

Már néhányszor tíz elemi esemény is olyan méretű állapotteret jelent, amelynek minden állapotát megvizsgálni kivárható időn belül nem lehet. Ezért valamilyen szempontok szerint ki kell tudni jelölni vizsgálatra a teljes tér egy részhalmazát. Erre az alábbi kapcsolók használatával nyílik mód:

`/M[N]` Az állapotok kiválasztása *mélység* szerint növekvő sorrendben. ((Hiba)mélység: az adott állapotban bekövetkezett események száma. A hibamentes állapot mélysége *0*, az egyhibásaké *1* stb.) A `/M` kapcsoló önmagában végiglépked az összes állapoton, de lekorlátozhatjuk indextartományok megadásával:

`/M0`	Csak a 0. (hibamentes) állapot.
`/M-99`	Az első száz (0, 1, …, 99 indexű) állapot.
`/M1-10,30+5,50`	Az 1, 2, …, 10, 30, 31, …, 35, 50 indexű állapotok.
`/M1000-`	Az 1000. és nagyobb indexű állapotok.

A `/M` kapcsoló után és a tartomány(ok) kijelölésében ne használjunk szóközt! A kiértékelés adatvesztés nélkül megszakítható `Ctrl-C`-vel.

`/V[N]` Az állapotok kiválasztása *valószínűség* szerint növekvő sorrendben. Itt is a fenti tartomány megadási szabályokat alkalmazhatjuk.

`/@SET` Egy állapot kiválasztása, melyben a `@SET` halmaz eseményei következnek be. Ha a halmazban vannak olyan elemek is, melyekhez nem tartozik esemény, akkor azokat a program figyelmen kívül hagyja. Ha a halmaz üres, akkor a hibamentes állapotot jelöljük ki.

A fenti kapcsolók közül csak az utoljára megadott hatása érvényesül, melynek jelentése az alábbiakkal tovább finomítható:

`/m>=M` A megadott *M* egész számnál kisebb mélységű állapotokat átugorjuk.

`/m<=M` Az *M*-nél nagyobb mélységű állapotokat átugorjuk.

`/ms<=M` Az *M*-nél nagyobb mélységű első állapot előtt leáll a kiértékelés.

`/p>=P` A *P* valós értéknél kisebb valószínűségű állapotokat átugorjuk.

`/ps>=P` A *P*-nél kisebb valószínűségű első állapot előtt leáll a kiértékelés.

`/pr>P` Ha az eddig nem kiértékelt állapotok valószínűsége már nem nagyobb, mint *P*, akkor leáll a kiértékelés.

2.11.4. Igények

Igényeknek nevezzük azokat a hálózatelemeket (elsősorban a link-eket, de a node-okat is), melyek működőképességét ellenőrizni akarjuk a vizsgált állapotokban. Megadni őket az állapotokat kijelölő kapcsolók után kell:

```
reli /M /ms<=1 link t mpx i Cap=*
```

A fenti parancs megvizsgálja a hibamentes és egyhibás állapotokban azokat az `mpx` link-eket, melyeknek van kapacitása, vagyis `Cap` info-ja.

Az állapotok jellemzője a valószínűségen kívül, hogy az igények összes kapacitásának mekkora hányada áll rendelkezésre. Egy igény rendelkezésre álló kapacitása a teljes kapacitásának és működőképességének [0..1] szorzata. A teljes kapacitást a `Cap` info definiálja, vagy ha ilyen nincs, akkor értéke *1*.

2.11.5. Eredmények

Az állapotokat jellemző mennyiségek: állapot index, hibamélység, valószínűség, működő kapacitás arány. Ennél részletesebb adatokat is kérhetünk az alábbi kapcsolókkal:

`/e` A 0. állapotban minden eseményt valószínűségével együtt felsorol, és megjelöli azokat, melyek bekövetkeztek. A többi állapotban csak a bekövetkezett eseményeket listázza.

`/d` A 0. állapotban minden igényt kapacitásával és működőképességével együtt felsorol, a többi állapotban csak a sérülteket, vagyis az *1*-nél kisebb működőképességűeket.

`/s` Az állapotok közé elválasztó üres sort nyomtat.

`/i` Beállítja az igények `DTRmin` és `DTRmax` infó-ját, ami az igénykiesési valószínűségnek a kiértékelt állapotok alapján számított alsó és felső becslése.

A kiértékelés végeztével az igények működőképessége az utolsó állapotra vonatkozóan benne marad az `Up` info-jukban.

A mélység és a valószínűség szerinti kiértékelés után még egy összesítő táblázatot is kapunk, melyben szerepel a kiértékelt állapotok száma és összes valószínűsége, valamint az igények kapacitáskiesésének átlagos relatív értéke. Ha az összes valószínűség *0.9*-nél nagyobb de *1*-nél kisebb, akkor helyette egy ennél *1*-gyel kisebb számot fogunk látni, ami a nem kiértékelt állapotok valószínűségének *-1*-szerese. Az igényekről is keletkezik egy lista, melyben megtalálható az igény átlagos meghibásodási valószínűsége.

2.12. OSPF

```
ospf [/LB[m] /C /A /Z /N /L]* [FSET]
```

Az `ospf FSET` parancs bővíti az OSPF elvezetéshez használt gráfot az `FSET`-ben szereplő, érvényes `OSPFarea` és `OSPFweightF/OSPFweightB` info-val rendelkező linkekkel és azok végpontjaival. A valóságos eszközök ugyan csak pozitív egész súlyokat értelmeznek, a programban megvalósított algoritmus azonban megenged 0 és nem egész súlyokat is. Ha jelezni akarjuk, hogy egy súly egy adott referencia sávszélességből számított default érték, akkor negatív előjellel adjuk meg.

A végpontok **nem** konvertálódnak `site` szintre. A gráf (bővítés előtti) törléséhez használjuk a `/C` kapcsolót. Az aktuális node és link halmaz lekérdezhető a `/N` és `/L` kapcsolókkal.

Az elvezetés kezdő- és végpontját a `/A` illetve a `/Z` kapcsolóval lehet megadni. Mindkét esetben csak az `FSET`-ben található első node-ot használjuk fel.

Miután definiáltuk a gráfot és a két végpontot, az elvezetést a paraméter nélküli `ospf` parancs listázza ki. Ha terhelésmegosztással (*load balancing*) is számolni akarunk, akkor használjuk a `/LB` kapcsolót, amit követhet még a routerek által maximálisan kezelt azonos súlyú kimenő linkek száma is (pl.: `ospf /LB4`). Ha az `/LB` kapcsoló paraméter nélkül áll, akkor nincs korlátozás a kimenő linkek számára vonatkozóan.

Ha a megbízhatósági analízis során van szükség `OspfB` blokkot tartalmazó igények kiértékelésére, akkor még az analízis indítása előtt definiálni kell az OSPF gráfot az `ospf [/C] FSET` paranccsal.

2.13. Gráfok

A hálózathoz kapcsolódóan több gráfot is létrehozhatunk, melyekre később nevükkel hivatkozhatunk. A név kötelezően '$' karakterrel kezdődik, amit tetszőleges számú alfanumerikus karakter követhet. Egy gráfot létrehozni, módosítani, törölni az alábbi alakú parancsokkal lehet:

```
$GRAF [GSW]* [[+]= [^]* [FSET]]
```

Pl. hozzunk létre egy `G1` nevű gráfot az `OPTIKAI` link-ekből:

```
$G1= link lk OPTIKAI
```

Adjuk hozzá a MIKRO link-eket is:

```
$G1+= link lk MIKRO
```

Lekérdezhetjük a gráf csomópontjainak és éleinek számát, ha parancsként csak a nevet írjuk, tehát pl. a `$G1` gráf esetén:

```
$G1
```

Az összes gráf lekérdezése egyetlen paranccsal:

```
graf
```

A gráf kitörléséhez az értékadás jobb oldalát hagyjuk üresen:

```
$G1=
```

Az összes gráf törlése:

```
graf clear
```

Abban az esetben is törlődik az összes gráf, ha kiadjuk bármelyik `erase` parancsot (lásd 2.7. Hálózatelemek törlése, 12. oldal).

Amikor egy link-et hozzáadunk egy gráf éleinek halmazához, akkor a csomópontok halmazához automatikusan hozzáadódik a két végpontja, amelyek a ***FlexPlanet***-ben lehetnek telephelyek, berendezések, kártyák vagy port-ok. Ha azonban nem akarunk port szintű modellen dolgozni, akkor használjuk az `FSET` előtt a '^' karaktert, és így a végpontok mindig a telephelyek lesznek:

```
$G2= ^ link lk OPTIKAI
```

Egy helyett több '^' karaktert is megadhatunk egyszerre (elválasztó szóközök nélkül), ami azt jelenti, hogy a telephely (^) helyett a berendezést (^^), kártyát (^^^), … tekintjük a link végpontjának. A konverzió csak „fölfelé" működik, vagyis pld. az alábbi sor a port-on való végződéseket helyettesíti a megfelelő berendezéssel, de a telephelyen való végződéseket változatlanul hagyja, még akkor is, ha az adott telephelyen csak egy berendezés van:

```
$G2= ^^ link lk OPTIKAI
```

A gráfok elemeit a program megjelöli egy-egy `$` típusú info-val, ahol az info értéke a gráf neve. Ha pl. egy link eleme a `$G1` és a `$G2` gráfnak is, akkor lesz egy `$=$G1` és egy `$=$G2` info-ja is.

Gráfokat leginkább utak kereséséhez vagy tervezéséhez használunk, mikor is fontos kérdés, hogy honnan hova akarunk eljutni. Többnyire a kiindulás és a cél is egy-egy konkrét csomópont, de bármelyik lehet pontok halmaza is, pl. ha olyan kérdésre keressük a választ, hogy melyik a legrövidebb út *bármelyik* gödöllői pontból *bármelyik* hatvani pontba. Az általánosság szem előtt tartása végett tehát célszerű a lehetséges kiindulási pontokat és a végpontokat is egy-egy halmazként kezelni, és a gráf egyéb paramétereivel együtt nyilvántartani. Ezeket a halmazokat a `/A` és a `/Z` kapcsolóval tudjuk megadni:

```
$G2 /A = node t site nn Godollo*
$G2 /Z = node t site nn Hatvan*
```

Miután adott a kiindulás és a cél, tervezhetünk utat:

```
$G2 /R1 /p
```

A `/R1` kapcsoló a minimálút tervezésére utasít, a `/p` pedig az eredmény listázására. Ha a végpontok közötti összes, egyformán minimális költségű utat meg akarjuk keresni, akkor használjuk a `/R1n` kapcsolót.

Két független, összegégében minimális utat a `/R2` vagy `/R2n` kapcsolókkal tervezhetünk. Az előbbi élfüggetlen, az utóbbi pedig pontfüggetlen utakat keres. Ha a `/R` kapcsolók közül többet is megadunk, akkor egy lépésben mindegyik tervezés eredményét megkapjuk.

A megtervezett utak alapján új linkek hozhatók létre:

```
@R= mpx r $G2
```

A parancs szintaktikáját részletesen a 2.6. fejezet (Új hálózatelemek felvétele, 11. oldal) tárgyalja. Az `r` kapcsoló annyi új link-et hoz létre, ahány megtervezett utat tárol a `$G2` gráf,

és ezután törli is $G2-ből ezeket az utakat. A felhasználatlan utakat törölhetjük a gráfból a /RC kapcsolóval.

Egy lépésben elvégezhető a tervezés több kiindulási és végpont között, ha a /A és /Z kapcsolók helyett a /AA illetve /ZZ kapcsolót használjuk. Pl. az alábbi parancsokkal kilistázhatjuk az utakat *mindegyik* gödöllői pontból *mindegyik* hatvani pontba:

```
$G2 /AA = node t site nn Godollo*
$G2 /ZZ = node t site nn Hatvan*
$G2 /R1 /p
```

Az *A* és a *Z* oldalon egymástól függetlenül használhatjuk a kapcsolók egy- és kétbetűs változatát. Ha az = operátor helyett +=-t írunk, akkor mindenképpen új halmazt hozunk létre. A következő példa utakat tervez *bármelyik* gödöllői pontból *mindegyik* hatvani pontba *és bármelyik* veresegyházi pontból *mindegyik* hatvani pontba:

```
$G2 /A = node t site nn Godollo*
$G2 /A += node t site nn Veresegyhaz*
$G2 /ZZ = node t site nn Hatvan*
$G2 /R1 /p
```

Az *A* illetve *Z* oldali halmazok lekérdezéséhez használható a /a illetve /z kapcsoló. A teljes gráfot (csomópontok és élek halmaza) listázhatjuk különböző részletességgel a /f, /g és /t kapcsolókkal.

Minimálút tervezésnél fontos szempont, hogy mit tekintünk az élek súlyának. Ha nem adunk meg semmit, akkor az élek „súlytalanok” lesznek, vagyis nincs mit minimalizálni, és az eredményül kapott útról csak annyit mondhatunk, hogy *egy lehetséges út* a kiindulás és a cél között. A /Hop, /Len és /Cap kapcsolók lehetővé teszik, hogy hop-szám, hossz vagy szabad kapacitás szerint keressünk minimálutat. Egyszerre több *tulajdonság* is megadható, ilyenkor lényeges a megadás sorrendje:

```
$G2 /Len /Cap /R1 /p
```

A fenti példában a legrövidebb utat keressük, azonban két azonos hosszúságú él közül azt fogjuk választani, amelyiken több a szabad kapacitás. A felsorolt tulajdonságokat sorrendjükkel együtt a gráfok megjegyzik, és ez a beállítás mindaddig érvényben marad, amíg ezt az adott gráfra vonatkozó paranccsal meg nem változtatjuk.

Az egyes élek súlyait a program a Hop, Len és Cap típusú info-kból veszi:

```
$G3 /Len /Cap = link lk OPTIKAI
$G3 /Hop += link lk MIKRO
```

$G3 gráfban az optikai link-ek hosszát és szabad kapacitását a megfelelő info-k határozzák meg, hop-számuk viszont akkor is 0 lesz, ha van ilyen info-juk. A mikro élek hossza 0 lesz, a hop-szám az info-ból derül ki, szabad kapacitásuk pedig a default érték, vagyis -1 lesz, ami a végtelent jelenti.

2.14. Listák

A listázó parancsok többsége elé (ahol ezt jelöljük is), opcionálisan írhatunk egy ’#’ karaktert, aminek hatására a program csak azt írja ki, hogy hány tételt tartalmazna a parancs ’#’ karakter nélküli változatának kimenete. Pl. a # node nn budapest* parancs kiírja a budapesti

pontok darabszámát, a `node nn budapest*` alak viszont valamennyit ki is listázza az aktuális lista formátumnak megfelelően.

2.14.1. Dimenziók

```
[#] dim
```

2.14.2. Szintek

```
[#] {level | szint} [* | *NEV | nNEV | kKAT | dDIM | D | S | cCSNEV]
```

`*`: Összes szint.

`*NEV`: Csak a `NEV` nevű szint.

`nNEV`: Azon szintek, melyek nevére illeszkedik `NEV`. (Ebben az esetben a jelölés arra utal, hogy `NEV` közvetlenül az 'n' karakter után kezdődik, és a sor végéig tart, tehát nincsenek határoló zárójelek vagy egyéb karakterek.)

`kKAT`: Azon szintek, melyek kategóriájára illeszkedik `KAT`.

`dDIM`: Azon szintek, melyek dimenziójára illeszkedik `DIM`.

`D`: A duplex szintek.

`S`: A szimplex szintek.

`cCSNEV`: Azon szintek, melyeknek van olyan csatornája, aminek a nevére illeszkedik `CSNEV`.

2.14.3. Sablonok (node template-ek)

```
[#] {templ | sabl} [* | *ID | nNEV | kKAT | pPOZ | iSNID | sSNNEV | lSLNEV]
```

`*`: Összes sablon.

`*ID`: Csak az `ID` azonosítójú sablon.

`nNEV`: Azon sablonok, melyek nevére illeszkedik `NEV`.

`kKAT`: Azon sablonok, melyek kategóriájára illeszkedik `KAT`.

`pPOZ`: Azon sablonok, melyeknek van olyan pozíciója, aminek a nevére illeszkedik `POZ`.

`iSNID`: Azon sablonok, melyeknek van olyan pozíciója, amibe `SNID`-re illeszkedő azonosítójú sablon dugható.

`sSNNEV`: Azon sablonok, melyeknek van olyan pozíciója, amibe `SNNEV`-re illeszkedő nevű sablon dugható.

`lSLNEV`: Azon sablonok, melyeknek van olyan pozíciója, amibe `SLID`-re illeszkedő nevű szint dugható.

2.14.4. Adott azonosítójú elem listázása

```
[#] *ID
```

2.14.5. Node-ok

```
[#] node [* | *ID | [FILT]*]
```

*: Összes node.

*ID: Csak az ID azonosítójú node-ok.

FILT: Csak a szűrőfeltételeknek megfelelő node-ok.

2.14.6. Link-ek

```
[#] link [* | *ID | [FILT]*]
```

*: Összes link.

*ID: Csak az ID azonosítójú link-ek.

FILT: Csak a szűrőfeltételeknek megfelelő link-ek.

2.14.7. Telephelyek

```
[#] site [* | FSET]
```

*: Összes telephely.

FSET: Az FSET-ben szereplő telephelyek.

2.14.8. Nyomvonal

```
path {l | g|G | p | r}* LINK_FSET
```

A parancs segítségével a LINK_FSET-ben szereplő elemek nyomvonalát listázhatjuk vagy módosíthatjuk. LINK_FSET egy olyan FSET, melyben csak link-ek lehetnek. Az alábbi kapcsolók közül legalább egyet meg kell adni. Több kapcsoló használata esetén azokat egybe kell írni, tehát nem szerepelhet köztük szóköz. (Pl.: `path rp link t mpx`)

g: Link hossza és a nyomvonal értékelése. A hosszt két értékkel jellemezzük:

- glen: A link rekurzív kifejtésének „alján” lévő gráf élek összes hossza.
- elen: A link rekurzív kifejtésének „alján” lévő folytonossági hiány légvonalban mérve. Nem csak hibátlan nyomvonal esetén lehet 0, hanem akkor is, ha azonos koordinátájú telephelyek között hiányos a leírás.

A nyomvonal értékelése az alábbi minősítések közül a legsúlyosabb (a listában leghátul következő):

- OK: Hibátlan szakaszolás (csak a közvetlen szakaszokat vizsgálva).

EXT: A link-nek van „Ext” info-ja.

LEN: A link rekurzív kifejtésének „alja” gráf szinten nem folytonos. Gráf élnek tekintjük a `graf` típusú link-eket, és azokat, melyeknek van `Graf` típusú info-juk. Utóbbiaknál a link hossza az info értéke.

REKH: A (rekurzív) hierarchiában valahol van egy olyan link, melynek a kifejtésében önmaga is szerepel.

REKK: A link kifejtésében önmaga is szerepel. (Nem feltétlenül a közvetlen szakaszok között, lehet, hogy mélyebb szinten.)

ERR: A közvetlen szakaszolás nem folytonos.

NOSZ: Nincs végpont.

NOSA: Nincs kezdőpont.

NOSB: A kifejtés nem soros blokk.

NOHIE: Nincs kifejtés.

G: Az fentiek közül nem ellenőrzi a közvetlen szakaszolás folytonosságát (ERR hiba nem lehet.)

l: Az fentiek közül nem ellenőrzi a gráf szintű hosszt és folytonosságot (LEN hiba nem lehet.)

p: Nyomvonal listázása. A fenti ellenőrzések eredménye előtt kilistázza a közvetlen szakaszpontokat (telephelyeket) és szakaszokat (link-eket) is. A listában jelzi azokat a helyeket, ahol nem teljesül a telephely szinten értelmezett folytonosság. (Két berendezés közti link-et csak akkor kell leírni, ha azok különböző telephelyen vannak.)

r: Nyomvonal szakaszpontjainak és szakaszainak sorba rendezése. A kapcsoló megadása esetén a program megpróbálja kijavítani a közvetlen szakaszpontok és szakaszok hibás sorrendjét, és amelyik link-nél ez sikerült, annak az info listája végére illeszt egy `Ref=Reo` bejegyzést.

2.14.9. Statisztikák

```
stat [t|k|l]*[site | p|g|G | i|I|iITIP] [FSET]
```

Az FSET elemeiről készít darabszám szerinti összesítést. A lehetséges csoportosítási szempontok az alábbiak lehetnek:

t: Típus (`site, fsite, device, card, port, log, dem, mpx, res, graf, …`).

k: Kategória (sablon vagy szint).

l: Sablon vagy szint.

site: Node vagy link végpontjainak telephelye.

p: `path p` minősítés.

g: `path g` minősítés.

`G`: `path G` minősítés.

`i`: Info típus.

`I` Info típus és érték

`iITIP` Info típus és érték csak az `ITIP` nevű info-kra.

A `t`, `k`, `l` szempontok mindegyike a többitől függetlenül elhagyható vagy megadható, de ha megadjuk, akkor kötelezően a csoportosítási kifejezés elején kell álljanak. A további szempontok közül legfeljebb egyet lehet megadni. Ha egyetlen szempontot sem adunk meg, akkor csak azt kapjuk meg, hogy `FSET` mennyi node-ot és mennyi link-et tartalmaz. Ha az `FSET` paramétert elhagyjuk, akkor a hálózat összes node-jára és link-jére készül a statisztika.

2.14.10. Linkterhelések osztályonkénti bontásban

```
sumcap [FSET]
```

Összegzi `FSET` link-jeinek kapacitását a kifejtésükben szereplő linkeken, osztályonkénti bontásban. Az `FSET`-ben megadott igények kapacitását a `Cap`, osztályát a `Class` info határozza meg.

Az összesítő listában megjelenő oszlopok beállíthatóak a következő fejezetben ismertetett `col` paranccsal, a kifejtésben vizsgált link-ek köre pedig szűkíthető a `hr` és `hg` halmazokkal (lásd később a 29. oldalon).

2.14.11. Konfigurálható oszlopok

Az egyszerű, nem hierarchikus `List` típusú kimeneti formátum alapértelmezésben csak a listázott elemek azonosításához szükséges minimális adatokat tartalmazza, melyek azonban bővíthetőek a `col` paranccsal. Az additív oszlopokat sorrendjükkel együtt az utoljára kiadott parancs határozza meg. Ha vissza akarjuk állítani az alapértelmezett állapotot, akkor adjuk ki a `col` parancsot argumentum nélkül, vagy nyissunk egy új `List` kimenetet.

A `col` paranccsal beállított additív oszlopok kiíródnak az `XLS` típusú listák alapértelmezett oszlopai után is. Ezekre az additív oszlopokra azonban a `mask` parancs (lásd később a 28. oldalon) beállításai hatástalanok maradnak.

```
col [templ|level | node|site | xy|clus | tav|{g|e}len | path[g] | iITIP]*
```

`templ`: Sablon.

`level`: Szint.

`node`: Link végpontjainak teljes neve.

`site`: Node-nak vagy link végpontjainak telephelye.

`xy`: Node-nak vagy link végpontjainak koordinátái.

`clus`: Cluster (körzet).

`tav`: Link végpontjainak földrajzi távolsága.

`glen`: Hossz (lásd `path g` parancs.)

`elen`: Hiányzó hossz (lásd `path g` parancs.)

`path`: `path p` minősítés.

`pathg`: `path g` minősítés.

`iITIP`: Az `ITIP` típusú info értéke. Ha több ilyen is van, akkor az értékeket '&' karakterrel összefűzi.

2.14.12. Listázási sorrend

```
ord [[f]nev|tip|sn|sk|ll|lk|site|sa|sz|SA|SZ|x|y|clus|ca|cz|iITIP]*
```

A listákban a node-ok és a link-ek alapértelmezésben ID szerint rendezve jelennek meg. Ezzel a paranccsal módosíthatjuk a `node`, `link`, `site`, `info` és `path` parancsok listázási sorrendjét. Új kimeneti lista megnyitásakor mindig visszaáll a gyorsabb, alapértelmezett sorrend. Ugyanez elérhető a lista lezárása nélkül is az `ord` parancs argumentum nélküli alakjával.

`nev`: Node vagy link neve (string).

`fnev`: Node teljes neve (string).

`tip`: Node vagy link típusa (string).

`sn`: Node sablon neve (string).

`sk`: Node sablon kategóriája (string).

`ll`: Link szintjének neve (string).

`lk`: Link szintjének kategóriája (string).

`site`: Node telephelyének neve (string).

`sa`: Link kezdőponti telephelyének neve (string).

`sz`: Link végponti telephelyének neve (string).

`SA`: Link kezdőpontjának teljes neve (string).

`SZ`: Link végpontjának teljes neve (string).

`x`: Node X koordinátája (numerikus).

`y`: Node Y koordinátája (numerikus).

`clus`: Node körzete (string / numerikus).

`ca`: Link kezdőponti telephelyének körzete (string / numerikus).

`cz`: Link végponti telephelyének körzete (string / numerikus).

`iITIP`: Node vagy link ITIP nevű info-ja. (string / numerikus).

Több rendezési szempontot is felsorolhatunk szóközökkel elválasztva, melyek balról jobbra érvényesülnek. A (string) jelű mennyiségeket ABC szerint rendezzük, a (numerikus) jelűeket valós számként. Az utóbbiakra csökkenő rendezést is előírhatunk, ha a szempont neve elé egy '-' karaktert írunk. A (string / numerikus) jelű mennyiségeket alapértelmezésben ABC szerint rendezzük, de ha a szempont neve elé '+' vagy '-' karaktert írunk, akkor az növekvő vagy csökkenő numerikus rendezést jelent.

```
ord iClass –iCap +iHop nev
```

A fenti példában az elemek elsődleges rendezési szempontja a `Class` info (ABC szerint), azon belül előre kerülnek a nagyobb kapacitásúak (`Cap` info), azon belül a kisebb hop számúak (`Hop` info), és ha az is egyezik, akkor az elem neve dönt. (Ha esetleg még a név is megegyezne, akkor ID szerint rendezünk.)

2.14.13. Egyéb listázási paraméterek

Az `XLS` és `SLK` formátumú listák oszlopait és sorait szűrhetjük az alábbi parancsokkal. Egy beállítás addig marad érvényben, amíg azt a hozzá tartozó paranccsal meg nem változtatjuk. A parancsok argumentum nélkül törlik a nekik megfelelő beállítást, új lista megnyitásakor pedig mindegyik beállítás törlődik.

Oszlopok kiírásának engedélyezése és letiltása:

```
mask [[+|-]NN | MASK]*
```

`MASK`: A string *n.* karaktere az *n.* oszlop (mező) kiírását vezérli. Amelyik oszlopra hibás karaktert adunk meg, az úgy fog viselkedni, mintha 'x' tartozna hozzá. Ugyanez a helyzet azokkal az oszlopokkal is, amelyekhez semmilyen karakter nem tartozik, mert `MASK` túl rövid.

- `x`: Az oszlop kiírásának engedélyezése. A listázók bizonyos esetekben, ha ez az áttekinthetőséget növeli, és az oszlop tartalma nem változott az előző sorhoz képest, akkor a mezőt üresen hagyják.
- `0`: Mint 'x', de csak a mező utolsó szavát írja ki. (Széles oszlop esetén lehet hasznos, ha a lényeg az oszlop végén van.)
- `1..9`: Mint 'x', de csak a mező első adott számú szavát írja ki. (Széles oszlop esetén lehet hasznos, ha a lényeg az oszlop elején van.)
- `+`: Az oszlop kiírásának engedélyezése, de az 'x' vezérléstől eltérően az ismétlődőseket is mindig kiíratjuk.
- `-`: Az oszlop kiírásának letiltása. (Nem üres mező nyomtatása, hanem teljesen kimarad az oszlop.)

`NN`: Oszlopindex (1-től indul). A következő `MASK` karakterek ettől a pozíciótól értendőek. A kisebb indexű pozíciók karakterei nem változnak, illetve ha még nem definiáltak, akkor 'x' karakterek lesznek. Ez utóbbi szabály megváltoztatható, ha `NN` elé '+' vagy '-' előjelet is írunk, ugyanis ekkor az adott előjel karakterrel töltődnek fel a még nem definiált oszlopvezérlő pozíciók.

Hierarchiamélység szerinti szűrés:

```
hm [HMAX]
```

`HMAX`: Hierarchikus listák maximális mélysége. Pl. linkek kifejtésénél a `hm 0` beállítás esetén csak a link-ek listázódnak, kifejtésük nem. A `hm 2` azt jelenti, hogy csak a link-eket, a közvetlen szakaszolásukat és még egy szintet akarunk látni. Speciális jelentése van a `graf` kulcsszónak, ami a linkek listázásánál csak a link szintet (`hm 0`) és a hierarchia legalját (általában ez a gráf szint) engedi listázni.

Oszlop tartalma szerinti szűrés:

```
hc [NN [STR]]
```

Csak azokat a sorokat engedi kiírni, amelyek NN. mezőjére illeszkedik STR. Ha NN=0, akkor *valamelyik* mezőre kell illeszkednie STR-nek.

Link-ek kifejtésének listázásakor megadhatjuk, hogy csak egy bizonyos halmazban szereplő elemek jelenjenek meg a listában:

```
hr [@SET]
```

Definiálhatjuk azoknak az elemeknek a halmazát, melyeket listázáskor a kifejtés „aljának” (gráf szint) kell tekinteni, akkor is, ha egyébként van hierarchiájuk:

```
hg [@SET]
```

2.15. Plugin-ok futtatása

```
LIBNAME.dll
```

Kilistázza a megadott nevű plugin függvényeit. A .dll kiterjesztést ki kell írni, mert ebből derül ki, hogy a parancs plugin hívás.

```
LIBNAME.dll FUNCNAME ?[HLP]
```

Kilistázza, hogy a FUNCNAME függvény milyen argumentumokat vár. Ha a kérdőjel után egy kifejezést is megadunk, akkor azokat a függvénybe épített rövid tájékoztatókat is kiírja, melyek címére illeszkedik a megadott minta. Valamennyit megjeleníthetjük az alábbi paranccsal:

```
LIBNAME.dll FUNCNAME ?*
```

```
LIBNAME.dll FUNCNAME [ARG]*
```

Meghívja a FUNCNAME függvényt a felsorolt argumentumokkal. Az argumentumokat szóköz karakterek határolják, ezért ha valamelyik argumentum maga is tartalmaz ilyen karaktert, akkor azt az argumentumot idézőjelek közé kell zárni. Az üres stringet a "" jelenti.

2.16. Egyéb parancsok

```
echo
```

Bekapcsolja az echo üzemmódot, vagyis minden parancs kiírja magát a standard outputra, ami batch jellegű feldolgozásoknál lehet hasznos. Kikapcsolni csak úgy lehet, ha megnyitunk egy újabb parancs-fájlt.

`dos`

Listázáskor az ékezetes karaktereket DOS-os formátumban jeleníti meg. (Ez pl. akkor hasznos, ha a DOS prompt képernyőjére listázunk.) Az üzemmód kikapcsoláshoz új kimeneti listát kell nyitni.

`child`

Alapértelmezésben az `XLS` és `SLK` formátumú listákban a node-ok hierarchiája a legfelső szinttől (ami általában a telephely) a közvetlen beültetett szintig jelenik meg. A `child` parancs ezt úgy módosítja, hogy a listába a node alatt levő összes elem is kiírásra kerül, rekurzívan a hierarchia aljáig. (Ez akkor lehet pl. hasznos, ha egy berendezés összes port-ját, és nem csak a kártyáit akarjuk listázni.) A parancs ismételt kiadása visszaállítja az alapértelmezett működést.

`{config | ?}`

Kiírja az aktuális kimeneti fájlba az érvényben lévő `Read, echo, List (XLS, SLK, XML, …), dos, child, col, mask, hm, hc, hr, hg` beállításokat.

`time [ID [0]]`

Argumentum nélkül kiírja a program indítása óta eltelt időt. A `time ID 0` forma létrehoz egy `ID` nevű időmérőt, és lenullázza. A nullázástól eltelt időt lekérdezni a `time ID` paranccsal lehet. `ID` csak alfanumerikus és aláhúzás (’_’) karakterekből állhat, a kis- és nagybetűk azonosnak számítanak, egyébként tetszőleges számú és nevű időmérőt létrehozhatunk.

A `Read` parancs kiadásakor automatikusan létrejön (0 kezdőértékkel) az „RD” nevű időmérő, a `Write` parancs kiadásakor pedig a „WR”. Ezenkívül ha engedélyezzük a „+TIMETXT” üzenetet a **`network.ini`**-ben, akkor minden egyes ***FlexPlanet*** fájl beolvasásának és kiírásának az idejét is kiírja az *FPCom*.

`{print | lf} [MSG]`

Kiírja az `MSG` üzenetet és egy soremelés karaktert az aktuális kimenetre.

`FLEXPLAN_COMMAND_FILE`

Ha létezik az adott nevű fájl, akkor beolvassa, és végrehajtja a benne szereplő parancsokat. A parancsfájlok hívása tetszőleges mélységben egymásba ágyazható.

Az alapértelmezett kiterjesztés az **`.fpc`**, amit automatikusan a megadott név után illeszt, ha nem adtunk meg más kiterjesztést. Ha a névben nincs ’:’ karakter, valamint nem "/", "./", "../" stb. stringgel kezdődik, akkor a név elé fűzi az `FPCDIR` környezeti változó tartalmát is.

`!OS_COMMAND`

Rendszerparancs hívás.

`end | exit | !`

Kilépés a programból.

2.17. Command line

Az *FPCom* indításakor megadott command line argumentumokat a program parancsként vagy parancsfájlként értelmezi. Egynél több parancs esetén köztük a ';' elválasztó karaktert kell használni.

Például olvassuk be a **DEMO** könyvtárban lévő munkaterületet, listázzuk ki az összes linket a képernyőre, majd lépjünk ki a programból:

```
FPCom Read DEMO; link *; !
```

3. Info típusok

Info típus	Info érték	Leírás
Clus	*n*, Res	Lásd: 2.7. Hálózatelemek törlése, 12. oldal.
DTR Up		Lásd: 2.11. Megbízhatósági analízis, 17. oldal.
OSPFarea OSPFweightF OSPFweightB		Lásd: 2.12. OSPF, 20. oldal.
Hop Len Cap $	$GRAF	Lásd: 2.13. Gráfok, 20. oldal.
Graf Ref Ext	Reo	Lásd: 2.14.8. Nyomvonal, 24. oldal.
Class		Lásd: 2.14.10. Linkterhelések osztályonkénti bontásban, 26. oldal.