

# System

Sisältää käyttöjärjestelmään ja laitteistoon liittyviä funktioita.

## **System.base64encode (strTxt)**

## **System.base64decode (strTxt)**

*strTxt*      *Tulostettava teksti*

Koodaa ja dekodaa tekstiä tai binääridataa base64 -muotoon. Kyseistä koodaustapaa käytetään usein binäärimuotoisen datan siirtämiseksi vain tekstimuotoa tukevan linjan ylitse – usein ascii 7-bit.

Paluuarvot

Palauttaa koodatun tai dekodatun datan.

Esimerkki:

```
-- Lue binääri dataa tiedostosta ja koodaa se
local bin = file:read ("*a")
local coded = System.base64encode(bin)
```

## **System.shell (strCmd)**

*strCmd*      *Suoritettava komento*

Suorittaa komennon järjestelmän shellissä (synkronisesti, eli kutsuva käyttöjärjestelmän prosessi odottaa että komento on suoritettu kokonaan) ja palauttaa komennon stdout:iin tulostaman tekstin riveittäin taulukossa. Palautettu taulukko on numeraalisesti indeksoitu 1 eteenpäin, ja yksi tulosteen rivi vastaa yhtä taulukon riviä.

Palauttaa *nil* jos komentoa ei voitu suorittaa, ja tyhjän taulukon jos komento ei tulosta mitään stdout:iin.

Komento palauttaa toisena paluuarvona suoritettun komennon paluuarvon, eli n.s. exit code:n.

Esimerkki:

```
-- hakemiston tiedostot
-- huom! että kaksi ensimmäistä aina . Ja ..
-- ne pitää ohittaa
```

```
local dir, ec = System.shell ("ls /tmp/*")
if #dir > 2 and ec == "0" then
    for i = 3, #dir do
        print (dir[i])
    end
end
```

```
end
end
```

### **System.log (prio, strCmd)**

**prio**            Loki merkinnän kiireellisyys. Kokonaisluku välillä 0 .. 7 jossa 0 vähiten kiireellinen (debug).

**strCmd**        Tulostettava teksti

Kirjoittaa merkinnän järjestelmä lokiin.

Paluuarvot

*true* mikäli onnistuu, muuten *nil*.

Esimerkki:

Slc.log(0, "This is my system log info")

### **System.nanosleep (intDly)**

**intDly**        Viive nanosekunteina

Vastaa POSIX kutsua nanosleep(). Asettaa kutsuvan säikeen uneen (minimissään) annetuksi ajaksi.

Esimerkki:

-- Plc ohjelma odottaa 1 millisekunnin

Slc.nanosleep (1000000)

### **System.nanotimer ()**

*Ei argumentteja*

Palauttaa järjestelmän tarkkuusajastimen nykyisen arvon. Ajastin on laskuri (uint64), joka alkaa juosta 0:sta kun käyttöjärjestelmä käynnistyy. Sen laskee nanosekuntteja, ja sen todellinen tarkkuus on usein noin 40 nanosekunnin luokkaa (ARM cortex A8).

Esimerkki:

print ("Current hires timer value: ".. System.nanotimer () )

### **System.parseUrl (strUrl)**

**strUrl**        Url teksti

Parsii URL merkkijonon, ja palauttaa URL eri kentät talukossa.

Esimerkinä:

URL: <https://areena.yle.fi/tv/ohjelmat/sarjat?t=uusimmat>

Parsittu taulukko:

```
URL_taulukko = {  
  protocol = "https",  
  user = "",  
  pass = "",  
  address="areena.yle.fi",  
  port="",  
  fullpath="/tv/ohjelmat/sarjat",  
  path={"tv", "ohjelmat", "sarjat"},  
  paramstring="t=uusimmat",  
  params={t="uusimmat"}  
}
```

Toinen esimerkki:

URL:

http://name:secret@www.google.com:8080/this/is/path/file.xml?param1=1&meaning=42

Parsittu taulukko:

```
{  
  protocol = "http",  
  user = "name",  
  pass = "secret",  
  address = "www.google.com",  
  port = "8080",  
  fullpath = "/this/is/path/file.xml",  
  path = {"this", "is", "path", "file.xml" },  
  paramstring = "param1=1&meaning=42",  
  params = {  
    param1 = "1",  
    meaning = "42"  
  }  
}
```

Virhetilanteessa funktio voi palauttaa arvon *nil*.

Esimerkki:

```
local parsed = System.parseUri ( urlString )  
if parsed.protocol == "http" then  
  -- Handle http request  
end
```

**System.encodeUri (strData)**

## **System.decodeUrl (strData)**

*strData*      *Tulostettava teksti*

Kun käsitellään URL tekstejä, on itse lokaattorin lopussa annettavien parametrien arvot koodattava niin sanotulla prosentti koodauksella. Näillä funktioilla voidaan tehdä parametrien arvojen prosenttikoodaus, ja dekodeaus.

Paluuarvot

koodattu tai purettu teksti.

Esimerkki:

-- Palauttaa meik%C3%A4%0A

local encoded = System.encodeUrl ("meikä")

## **System.serialize (data)**

*data*      *Serialisoitava data (merkkijono, luku, taulukko)*

Tekee lua objekteille niin sanotus sarjoituksen, eli serialisoinnin.

Tämä muunnos tarkoittaa että objekti muutetaan takaisin lähdekoodi -muotoon. Tätä muunnosta tarvitaan esimerkiksi silloin kun lua taulukko halutaan siirtää tcp yhteyden ylitse toiseen laitteeseen, tai vain IPC-kanavan lävitse laitteen muistissa toiseen prosessiin, tai tallentaa tiedostoon.

Serialisoitu objekti on helppoa palauttaa takaisin ohjelmassa käsiteltäväksi, koska se voidaan "ajaa" normaalissa lua tulkissa.

Vastaa suurinpiirtein javascriptin JSON.stringify() kutsua.

Tämä funktio on käyttökelpoinen paitsi tiedonsiirrossa ja tallentamisessa, myös ohjelmien debuggauksessa, koska miltei mikä tahansa objekti voidaan tulostaa tekstimuodossa vaikkapa konsoliin ohjelmoijan tarkasteltavaksi.

Paluuarvot

Annettu objekti tekstimuodossaan merkkijonona, tai nil mikäli kutsuttiin virheellisillä arvoilla.

## **System.importCSV (strCsv [, strDlm] )**

*strCsv*      *Tab eroteltu data*

*strDlm*      *Sarakeet erottava merkki, oletus on \t eli tabulaattori*

Tämä funktio luo CSV muotoisesta – tai oletusarvoisesti TAB-erotellusta tekstistä – taulukon ja palauttaa sen.

Nimestään huolimatta oletus sarake-erottimelle on '\t'.

Luotavan rivin nimen oletetaan löytyvän sarakkeesta, jonka nimi on "dataname", "pointname", "rowname", "datapoint" tai "keyname" – kaikki edellämainitut ovat synonyymejä.

Esimerkki:

```
csv = [[name    description    pv
data1      tunnit      7.0
data2      minuutit    15.0
data3      sekunnit    23.0
data4      paivat      1.0
]]
```

```
local d = System.importCSV (csv)
```

>> taulukon d sisältö:

```
d = {
  data1 = {description = "tunnit", pv = 7},
  data2 = {description = "minuutit", pv = 15},
  data3 = {description = "sekunnit", pv = 23},
  data4 = {description = "paivat", pv = 1}
}
```

### **System.archInfo ()**

Funktio palauttaa taulukossa laitteiston käyttöjärjestelmän ja prosessoriarkkitehtuurin.

Palauttaa taulukon, jossa rivit

os      Käyttöjärjestelmän nimi. Joitakin tavallisia arvoja ovat mm.

**Ubuntu, Builtroot, Debian.**

arch    Prosessoriarkkitehtuuri. Tyypillisiä arvoja ovat

**armv7l** (beagle bone) ja **x86\_64**

Esimerkki:

```
local d = System.archInfo (csv)
print ("Käyttöjärjestelmä: ".. d.os)
```

### **System.getNetworkInterfaces ()**

Funktio palauttaa taulukossa järjestelmän verkkosovittimet ja niiden asetukset.

### **System.pid ()**

Funktio palauttaa kutsuvan prosessin (yleensä slc engine) process ID (eli pid) numeron. Tämä on se tekninen tunnus, jolla käyttöjärjestelmä tunnistaa prosessit, ja jonka avulla voi lähettää mm. signaaleita prosessien välillä.

### **System.sendSig (pid, signal)**

*pid* (int) prosessin ID numero

*signal* (int) signaali joka lähetetään (kts. *posix singals*)

Funktion avulla voi lähettää signaalin käyttöjärjestelmäprosessien välillä. Palauttaa *true* onnistuessaan, ja *nil* mikäli kutsussa ilmeni virhe.

---

Revision #2

Created 25 May 2022 09:53:53 by Severi Hiltunen

Updated 10 May 2023 10:25:34 by Severi Hiltunen