Bendar, fylki, eintengdir listar

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-01-18

Ábending – man síður

- Skjölun um föll í C er að finna í svokölluðum `man` síðum (manual pages)
- Fylgir með flestum Linux og Unix kerfum
- Skipt í kafla: Kafli 2 – system calls; Kafli 3 – library calls.
- Uppfletting í skipanalínu:

```
hhg@hhg:~/t2$ man 2 read
hhg@hhg:~/t2$ man 3 strcpy
```

- Aðrir geta notfært sér HTML útgáfu af Linux man síðunum:

```
http://t2.hhg.to/man/index2.html
http://t2.hhg.to/man/index3.html
```

- Þegar það þarf að nota ný föll í skilaverkefnum þá set ég almennt hlekki yfir á viðeigandi man síður

strtoupper: Útgáfa 1

```c
void strtoupper(char *s) {
  while (s && *s) {
    *s = toupper(*s);
    s++;
  }
}

int main(void) {
  char s[] = "hello world";

  strtoupper(s);

  printf("%s\n", s);
}
```

Keyrsla:

```
hhg@hhg:~/t2$ ./q1
HELLO WORLD
hhg@hhg:~/t2$
```

strtoupper: Útgáfa 2

```c
void strtoupper(char *s) {
  while (s && *s) {
    *s++ = toupper(*s);
  }
}

int main(void) {
  char s[] = "hello world";

  strtoupper(s);

  printf("%s\n", s);
}
```

Keyrsla:

```
hhg@hhg:~/t2$ ./q2
HELLO WORLD
hhg@hhg:~/t2$
```

strtoupper: Útgáfa 3

```c
void strtoupper(char *s) {
  if (!s)
    return;

  while (*s++ = toupper(*s))
    ;
}

int main(void) {
  char s[] = "hello world";

  strtoupper(s);

  printf("%s\n", s);
}
```

Keyrsla:

```
hhg@hhg:~/t2$ ./q3
HELLO WORLD
hhg@hhg:~/t2$
```

Samantekt: Notkun * og ++ saman

`*p++` eða `*(p++)` – Algengast að sjá `*p++`

- Gildi segðarinnar er *p áður en það er hækkað
- p er hækkað eftir gildingu

`(*p)++`

- Gildi segðarinnar er *p áður en það er hækkað
- *p er hækkað eftir gildingu

`*++p` eða `*(++p)`

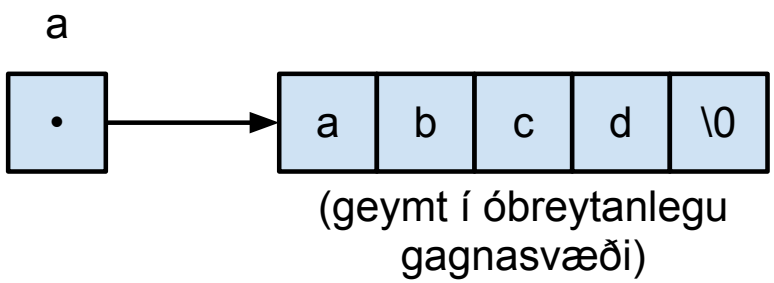
- p er hækkað fyrir gildingu
- Gildi segðarinnar er *p eftir hækkunina

`++*p` eða `++(*p)`

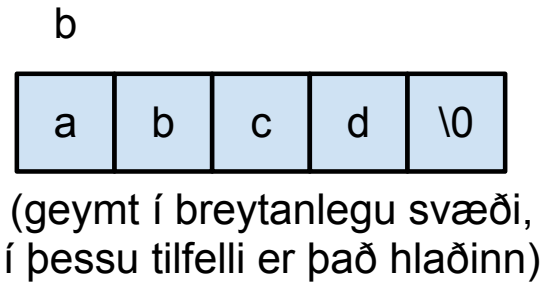
- *p er hækkað fyrir gildingu
- Gildi segðarinnar er *p eftir hækkunina

Strengjagildi (string literals)

```c
int main(void)
{
    char *a = "abcd";

    a[0] = 'x'; // crash
}
```




```c
int main(void)
{
    char b[] = "abcd";

    b[0] = 'x'; // ok
}
```



Minnissvæði

Þula (code):

- Breytist aldrei
- Hvorki úthlutað né skilað í keyrslu
- Inniheldur framkvæmanlegan forritstexta (vélamálskóða)

```
0x08048394 <+0>: push   %ebp
0x08048395 <+1>: mov    %esp,%ebp
0x08048397 <+3>: sub    $0x20,%esp
0x0804839a <+6>: movl   $0xa,-0x8(%ebp)
0x080483a1 <+13>: lea    -0x10(%ebp),%eax
0x080483a4 <+16>: mov    %eax,-0x4(%ebp)
0x080483a7 <+19>: movl   $0x14,-0x10(%ebp)
0x080483ae <+26>: lea    -0x18(%ebp),%eax
0x080483b1 <+29>: mov    %eax,-0xc(%ebp)
0x080483b4 <+32>: movl   $0x1e,-0x18(%ebp)
0x080483bb <+39>: movl   $0x0,-0x14(%ebp)
0x080483c2 <+46>: leave
0x080483c3 <+47>: ret
```

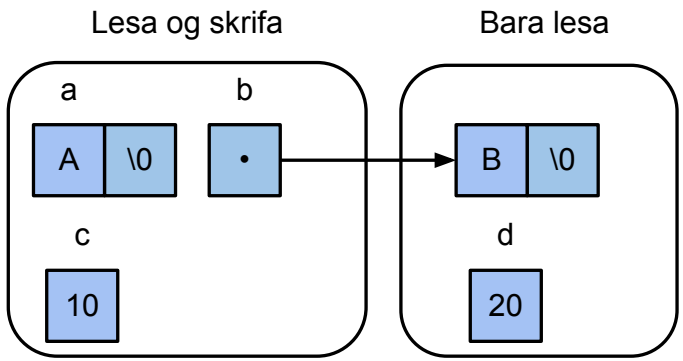
```
0x55  0x89  0xe5  0x83  0xec  0x20  0xc7  0x45
0xf8  0x0a  0x00  0x00  0x00  0x8d  0x45  0xf0
0x89  0x45  0xfc  0xc7  0x45  0xf0  0x14  0x00
0x00  0x00  0x8d  0x45  0xe8  0x89  0x45  0xf4
0xc7  0x45  0xe8  0x1e  0x00  0x00  0x00  0xc7
0x45  0xec  0x00  0x00  0x00  0x00  0xc9
```

Minnissvæði

Gögn (data):

- Hvorki úthlutað né skilað í keyrslu
- Inniheldur gögn fyrir víðværar (global) breytur og þær staðværar (local) breytur sem eru fastbundnar (static)
- Skiptist í tvö svæði: annað má bara lesa, hitt má lesa og skrifa

```
// Víðværar breytur
char a[] = "A";
char *b = "B";
int c = 10;
const int d = 20;

int main(void) {
  // ..
}
```



Minnissvæði

Hlaði (stack):

- Má breyta
- Úthlutað sjálfkrafa í keyrslu við innkomu í stef eða inn í bálk ({...}). Skilað sjálfkrafa þegar snúið er til baka.
- Inniheldur m.a. viðföng og staðværar breytur

```
int *f(void) {
  int n;

  return &n; // Slæm hugmynd
}

int main(void) {
  int *p = f();
}
```

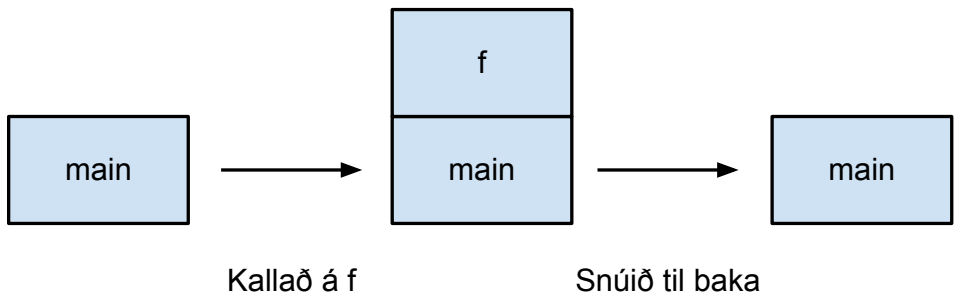
- Minni fyrir `n` er úthlutað sjálfkrafa við innkomu í f
- Minninu er skilað þegar snúið er til baka úr f
- Ófyrirsjáanleg virkni ef minnið er notað eftir að því er skilað

Minnissvæði

Hlaði (stack):

```
void f(int *p) {
  // ..
}

int main(void) {
  int n;

  f(&n);
  // ..
}
```

- Þetta er hinsvegar allt í lagi
- Minninu fyrir `n` er ekki skilað fyrr en main snýr til baka
- Má alveg senda benda á staðværar breytur "upp" í föll ofar á hlaðanum, en ekki "niður" til baka



Minnissvæði

Kös (heap):

- Má breyta
- Úthlutað með `malloc`, skilað með `free`
- Inniheldur minnissvæði sem sumir bendar vísa á, þ.e. sem hafa fengið gildi beint eða óbeint úr malloc. Athugið að bendarnir sjálfir geta verið geymdir í ýmsum breytum, ýmist í gagnasvæði, hlaða eða kös.

```
int main(void)
{
  char *p = malloc(100);
  free(p);
}
```

- `#include <stdlib.h>`
- Hrátt viðmót: gefur upp bætafjölda og færð bendi á minnisblokk til baka
- malloc skilar taginu `void *`, sem er ótagaður bendir

strtoupper: Útgáfa 4

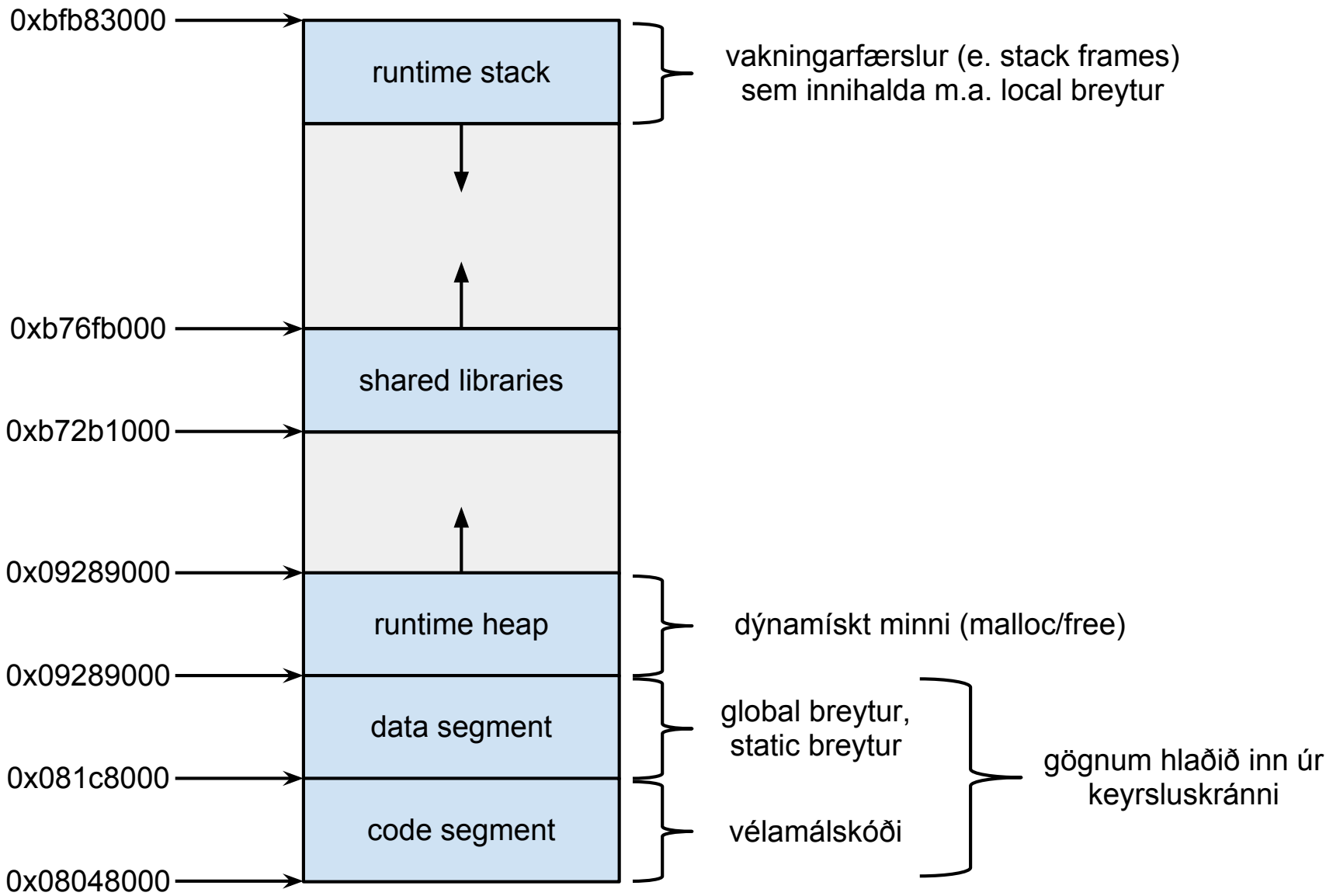
```c
char *strtoupper(char *s)
{
  if (!s)
    return NULL;

  char *p = malloc(strlen(s) + 1);
  char *w = p;

  while (*s)
    *w++ = toupper(*s++);

  *w = 0;

  return p;
}
```

Keyrsla:

```
hhg@hhg:~/t2$ ./q4
HELLO WORLD
hhg@hhg:~/t2$
```

```c
int main(void)
{
  char s[] = "hello world";

  printf("%s\n", strtoupper(s));
}
```

Sýndarminni í Linux



Fylki og bendar

Munið: Gildið á *a*, þar sem *a* er fylki, er í raun vistfangið á fyrsta stakinu í fylkinu.

Gerum ráð fyrir:

```
int a[10], *p, i = 2;
```

þá er eftirfarandi þrennt jafngilt:

```
p = a;
p[i];
```

```
p = a;
*(p+i);
```

```
p = a+i;
*p;
```

(`p[i]` er í raun þýtt yfir í `*(p+i)` á bakvið tjöldin.)

Geymslusvæði fylkis

- Í Java þá eru fylki alltaf geymd í kös
- Í C þá er hægt að geyma fylki á ólíkum stöðum, ræðst af því hvar og hvernig það er skilgreint

```
// Víðvær breyta, geymt í gagnasvæði
int a[100];

int main(void)
{
  // Staðvær breyta, geymt í vakningu á hlaða
  int b[100];
  // Fastbundin staðvær breyta, geymt í gagnasvæði
  static int c[100];
  // Minnisblokk úthlutað með malloc, fylki geymt í kös
  int *d = malloc(sizeof(int) * 100);
}
```

Fylki af bendum á strengi

```c
int main(void)
{
  char s[] = "hello world";
  char *v[32];

  v[0] = s;
  v[1] = s+3;
  v[2] = &s[3];
  v[3] = &s[6];

  printf("%s\n", v[0]);
  printf("%s\n", v[1]);
  printf("%s\n", v[2]);
  printf("%s\n", v[3]);
}
```

Keyrsla:

```
hhg@hhg:~/t2$ ./q10
hello world
lo world
lo world
world
hhg@hhg:~/t2$
```

struct

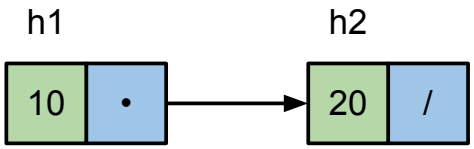
```c
struct person {
    char name[32];
    int age;
};

int main(void) {
    struct person t;
    struct person *p = &t;

    strcpy(t.name, "abc");
    t.age = 100;

    p->age += 1;
    (*p).age += 1;
}
```

- `struct` skilgreinir samsett gagnatag (eins og klasi í Java nema án aðferða)
- Vinnum með struct eins og hvaða annað tag:

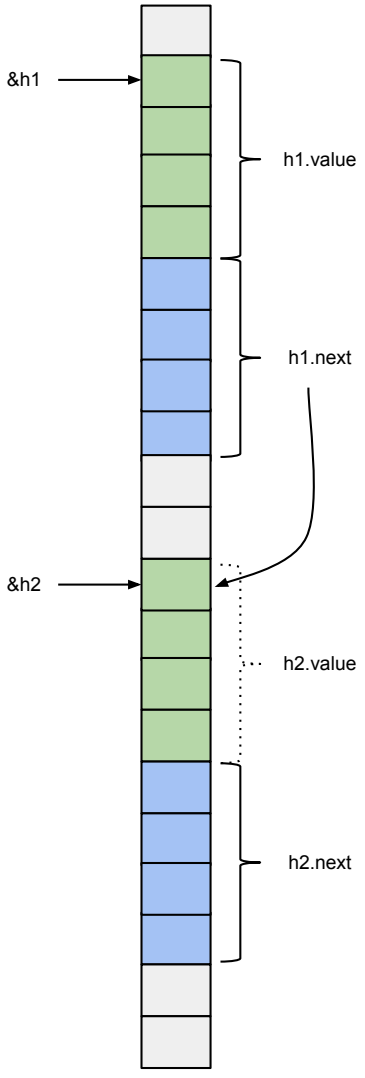
```c
int t1;
struct person t2;
```

- `t.x` sækir `x` úr `t`, þar sem `t` er struct gildi
- `t->x` sækir `x` úr `t`, þar sem `t` er bendir á struct gildi

Eintengdur listi

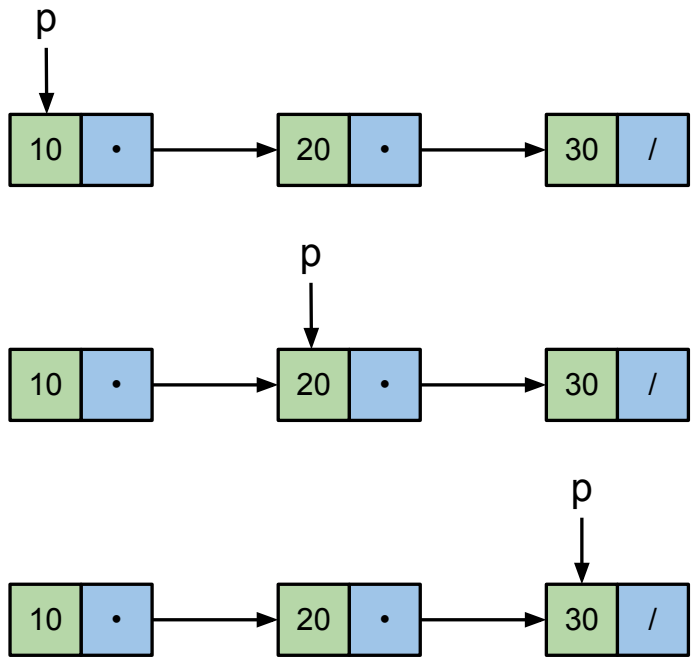



```
struct link {
    int value;
    struct link *next;
};

int main(void) {
    struct link h1, h2;

    h1.value = 10;
    h1.next = &h2;

    h2.value = 20;
    h2.next = NULL;
}
```



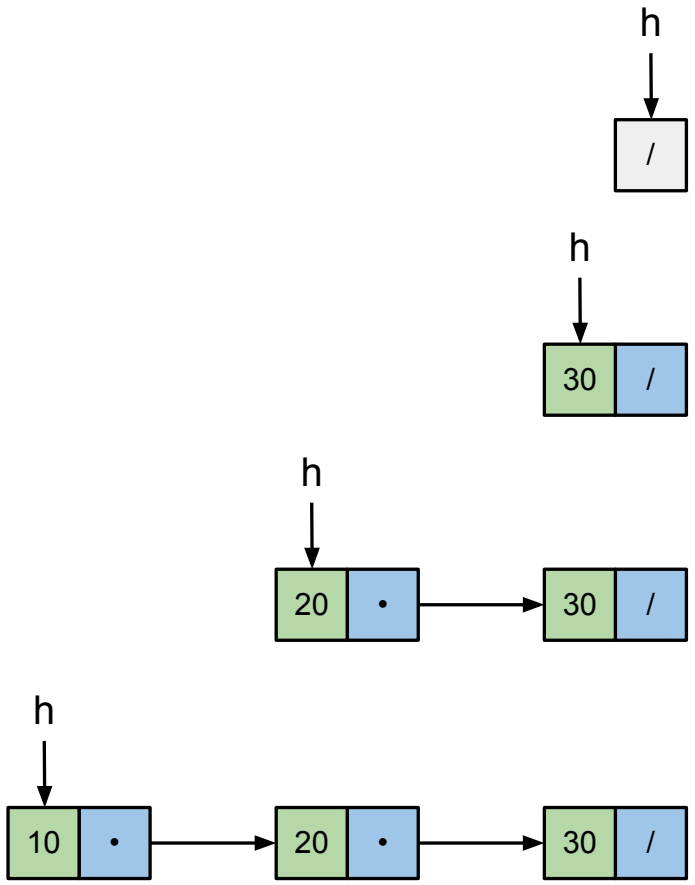
Göngum í gegnum eintengdan lista

```c
struct link
{
    int value;
    struct link *next;
};

void list_print(struct link *p)
{
    while (p != NULL)
    {
        printf("%d\n", p->value);

        p = p->next;
    }
}
```



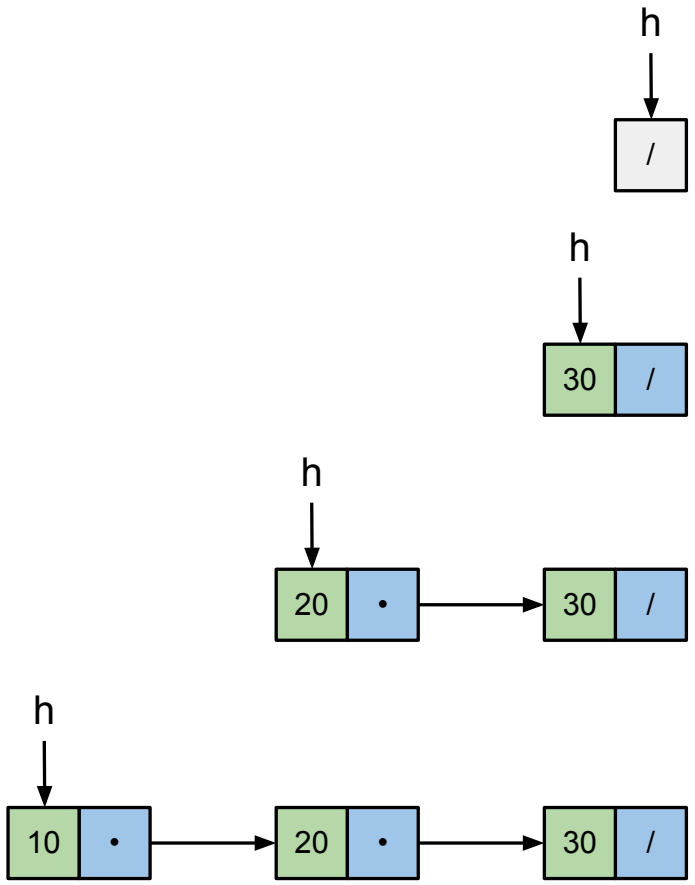
Smíðum eintengdan lista

```
struct link *list_add(struct link *h, int value) {
  struct link *p = malloc(sizeof(struct link));

  p->value = value;
  p->next = h;

  return p;
}

int main(void) {
  struct link *h = NULL;

  h = list_add(h, 30);
  h = list_add(h, 20);
  h = list_add(h, 10);

  list_print(h);
}
```



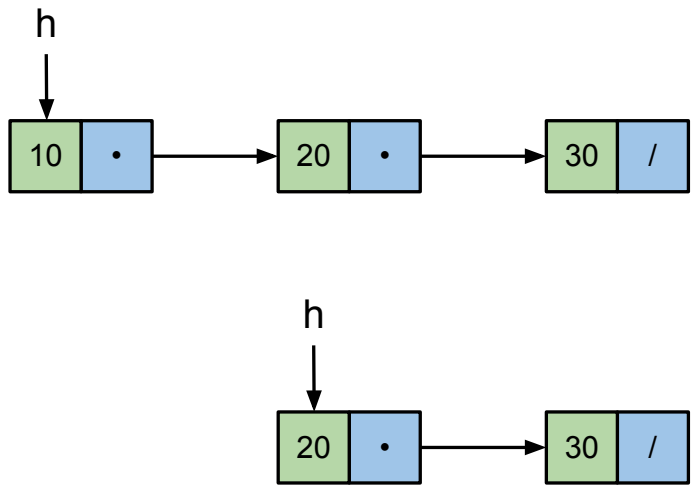
Smíðum eintengdan lista

```
void list_add(struct link **h, int value) {
  struct link *p = malloc(sizeof(struct link));

  p->value = value;
  p->next = *h;

  *h = p;
}

int main(void) {
  struct link *h = NULL;

  list_add(&h, 30);
  list_add(&h, 20);
  list_add(&h, 10);

  list_print(h);
}
```



Eyðum fremst úr eintengdum lista

```
struct link *list_remove(struct link *h)
{
  struct link *p = h->next;
  free(h);
  return p;
}

int main(void) {
  struct link *h = NULL;

  list_add(&h, 30);
  list_add(&h, 20);
  list_add(&h, 10);

  h = list_remove(h);

  list_print(h);
}
```



Eyðum fremst úr eintengdum lista

```
void list_remove(struct link **h) {
  struct link *p = *h;
  *h = p->next;
  free(p);
}

int main(void) {
  struct link *h = NULL;

  list_add(&h, 30);
  list_add(&h, 20);
  list_add(&h, 10);

  list_remove(&h);

  list_print(h);
}
```