ECE430.217 Data Structures

Abstract Priority Queues

Textbook: Weiss Chapter 6.1, 6.2

Byoungyoung Lee
https://compsec.snu.ac.kr
byoungyoung@snu.ac.kr

Lecture slides were prepared based on Douglas W. Harder's notes (**dwharder@alumni.uwaterloo.ca).**

Outline

This topic will:

- Review queues
- Discuss the concept of priority and priority queues
- Look at two simple implementations:
  - Arrays of queues
  - AVL trees
- Introduce heaps, an alternative tree structure which has better run-time characteristics

Background

We have discussed Abstract Lists

- Arrays, linked lists

We saw three cases which restricted the operations:

- Stacks, queues, deques

Then, we studied search trees: Abstract Sorted Lists

- Run times were generally $\Theta(\ln(n))$

We will now look :

- Priority queues
- Restriction on Abstracted Sorted Lists

Definition

With queues

- The order may be summarized by *first in, first out*

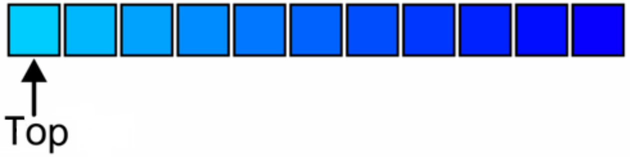
If each object is associated with a priority, we may wish to pop that object which has highest priority

With each pushed object, we will associate a nonnegative integer (0, 1, 2, ...) where:

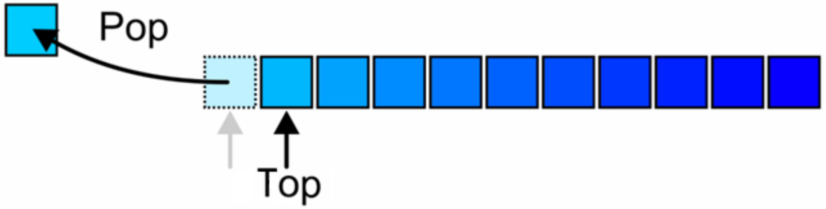
- The value 0 has the *highest* priority, and
- The higher the number, the lower the priority

Operations

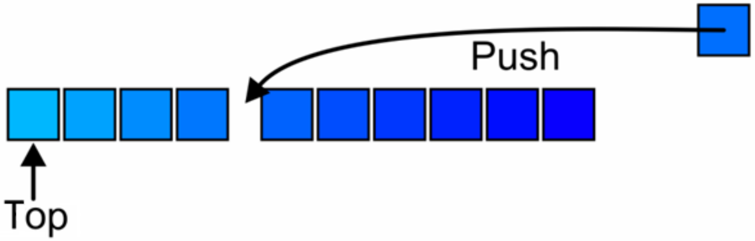
The top of a priority queue is the object with highest priority



Popping from a priority queue removes the current highest priority object:



Push places a new object into the appropriate place



Process Priority in Linux

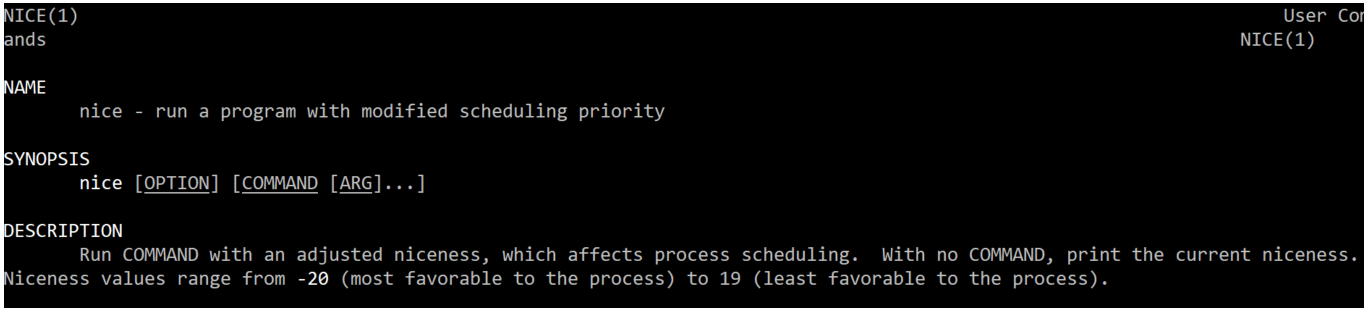
This is the scheme used by Linux, *e.g.*,

```
% nice -15 ./a.out
```

sets the priority of the execution of `a.out` as 15

The kernel will schedule processes according to the priority

$ man nice



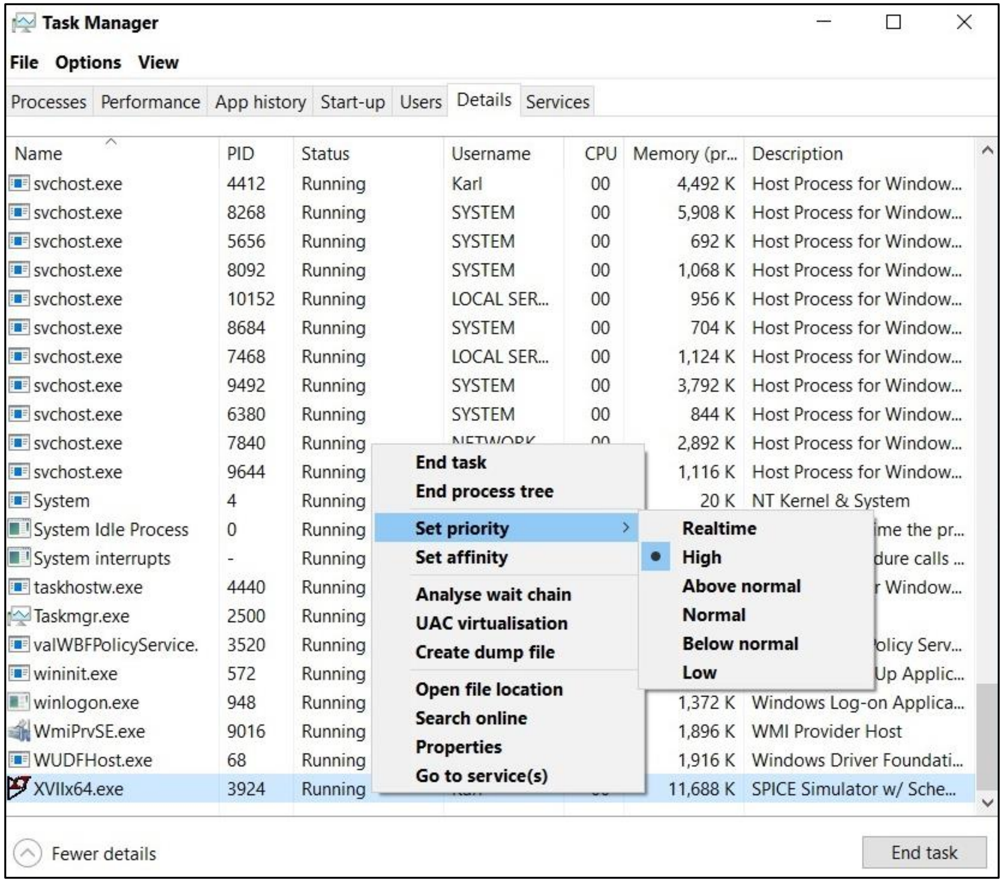
```
NICE(1)                                                                              User Cor
ands                                                                             NICE(1)

NAME
       nice - run a program with modified scheduling priority

SYNOPSIS
       nice [OPTION] [COMMAND [ARG]...]

DESCRIPTION
       Run COMMAND with an adjusted niceness, which affects process scheduling.  With no COMMAND, print the current niceness.
Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).
```

Process Priority in Windows



Name	PID	Status	Username	CPU	Memory (pr...	Description
svchost.exe	4412	Running	Karl	00	4,492 K	Host Process for Window...
svchost.exe	8268	Running	SYSTEM	00	5,908 K	Host Process for Window...
svchost.exe	5656	Running	SYSTEM	00	692 K	Host Process for Window...
svchost.exe	8092	Running	SYSTEM	00	1,068 K	Host Process for Window...
svchost.exe	10152	Running	LOCAL SER...	00	956 K	Host Process for Window...
svchost.exe	8684	Running	SYSTEM	00	704 K	Host Process for Window...
svchost.exe	7468	Running	LOCAL SER...	00	1,124 K	Host Process for Window...
svchost.exe	9492	Running	SYSTEM	00	3,792 K	Host Process for Window...
svchost.exe	6380	Running	SYSTEM	00	844 K	Host Process for Window...
svchost.exe	7840	Running	NETWORK	00	2,892 K	Host Process for Window...
svchost.exe	9644	Running			1,116 K	Host Process for Window...
System	4	Running			20 K	NT Kernel & System
System Idle Process	0	Running				ime the pr...
System interrupts	-	Running				dure calls ...
taskhostw.exe	4440	Running				r Window...
Taskmgr.exe	2500	Running				
valWBFPolicyService.	3520	Running				olicy Serv...
wininit.exe	572	Running				Up Applic...
winlogon.exe	948	Running			1,372 K	Windows Log-on Applica...
WmiPrvSE.exe	9016	Running			1,896 K	WMI Provider Host
WUDFHost.exe	68	Running			1,916 K	Windows Driver Foundati...
XVIIx64.exe	3924	Running	Karl	00	11,688 K	SPICE Simulator w/ Sche...

Implementations

Our goal is to make the run time of each operation as close to $\Theta(1)$ as possible

We will look at two naïve implementations using data structures we already know:

- Multiple queues—one for each priority
- An AVL tree

Multiple Queues

Assume there is a fixed number of priorities, say M

- Create an array of M queues
- Push a new object onto the queue corresponding to the priority
- Top and pop find the first empty queue with highest priority

Multiple Queues

The run times are reasonable:

- Push is $\mathbf{\Theta}(1)$
- Top and pop are both $\mathbf{O}(M)$

Unfortunately:

- It restricts the range of priorities

AVL Trees

We could simply insert the objects into an AVL tree where the order is given by the stated priority:

- Insertion is $\Theta(\ln(n))$
- Top is $\Theta(\ln(n))$
- Remove is $\Theta(\ln(n))$

There is significant overhead for maintaining both the tree and the corresponding balance

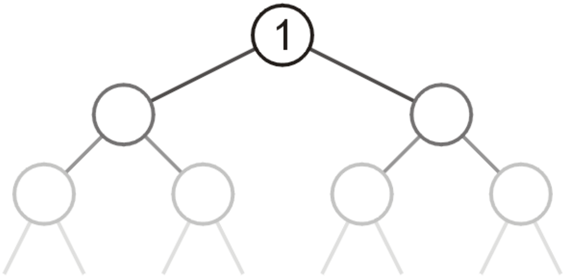
Better Idea: Heaps

Can we do better?

- That is, can we reduce some (or all) of the operations down to $\Theta(1)$?

The next topic defines a *heap*

- A tree with the top object at the root
- We will look at binary heaps
- Numerous other heaps exists:
    - d-ary heaps
    - Leftist heaps
    - Skew heaps
    - Binomial heaps
    - Fibonacci heaps
    - Bi-parental heaps



Summary

This topic:

- Introduced priority queues
- Considered two obvious implementations:
  - Arrays of queues
  - AVL trees
- Discussed the run times and claimed that a variation of a tree, a heap, can do better

References

Cormen, Leiserson, Rivest and Stein,
Introduction to Algorithms, The MIT Press, 2001, §6.5, pp.138-44.

Mark A. Weiss,
Data Structures and Algorithm Analysis in C++, 3rd Ed., Addison Wesley, 2006, Ch.6, p.213.

Joh Kleinberg and Eva Tardos,
Algorithm Design, Pearson, 2006, §2.5, pp.57-65.

Elliot B. Koffman and Paul A.T. Wolfgang,
Objects, Abstractions, Data Structures and Design using C++, Wiley, 2006, §8.5, pp.489-96